

09_博客 9-14周

09_博客 9-14周

Week 9-10

项目进展

代码分析

导入库

应用程序初始化

“开始游戏”按钮被点击的事件处理函数

“继续游戏”按钮被点击的事件处理函数

“显示结果”

“退出游戏”

输入参与游戏的总人数

点击“确定”按钮，读取当前玩家输入

读取玩家输入，若全部完成则显示游戏结果

若为直接调用，则运行程序

参考文档

week 11-12

工作进展

UI界面

数据库功能

操作

实现方法

UI界面

数据库保存

键盘事件的监听

下一阶段目标

week 13-14

工作进展

实现了用户登陆、注册功能

实现了远程游玩的功能

实现方法

服务器端：

客户端：

client.py

loginFrame.py

registerDialog.py

roomFrame.py

gameFrame.py

Week 9-10

项目进展

目前已经实现了黄金点游戏的基本功能，GUI用户界面也开发完成，但目前软件仅仅为本地客户端，接下来将进行GUI界面的优化，并实现C/S的运行模式。

代码分析

导入库

程序用python语言实现，用wxpython进行GUI设计

```
1 import wx
2 import math
3 from main import *
```

首先，创建一个wx.Frame对象。wx.Frame 是重要的容器widget。 wx.Frame类是其他widget的父类。它本身没有父类。为层次结构中的顶级widget。创建wx.Frame小部件之后，必须调用Show()方法才能在屏幕上实际显示。

```
1 class MyFrame(wx.Frame):
```

应用程序初始化

```
1     def __init__(self):
2         super().__init__(parent=None, title='黄金点数游戏',size = (450,300))
3         ##### 游戏数据
4         self.player_num = 0
5         self.golen_score = 0
6         self.player_score = []
7         self.player_input = []
8         self.tmpId = 1
9         self.maxId = -1
10        self.minId = -1
11        ##### 组件
12        panel = wx.Panel(self)
13        self.txt = TransparentStaticText(panel,-1,label = 'hello
world',pos=(10,10))
14        self.txt.Hide()
15        # self.txt1 = wx.StaticText(panel, -1, label='hello world', pos=
(10, 10))
16        self.strt_btn = wx.Button(panel, label='开始游戏', pos=(200, 125))
17        self.ctnu_btn = wx.Button(panel,label = '继续游戏',pos = (30,220))
18        self.show_btn = wx.Button(panel,label = '显示分数',pos = (180,220))
19        self.exit_btn = wx.Button(panel,label = '结束游戏',pos = (320,220))
20        self.ctnu_btn.Hide()
21        self.show_btn.Hide()
22        self.exit_btn.Hide()
23
```

```

24         self.player_numText = wx.StaticText(panel,-1,label='请输入玩家人
数',pos=(10,70),size=(-1,-1))
25         self.player_numTxcl = wx.TextCtrl(panel,-1,value='',pos=
(120,70),size = (100,20))
26         self.pncm_btn = wx.Button(panel,label = '确定',pos = (240,70),size=
(60,20))
27         self.player_numText.Hide()
28         self.player_numTxcl.Hide()
29         self.pncm_btn.Hide()
30
31         self.inputNotice = "请玩家"+str(self.tmpId)+"输入"
32         self.inputText = wx.StaticText(panel,-1,label=self.inputNotice,pos=
(10,70),size=(-1,-1))
33         self.inputTxcl = wx.TextCtrl(panel,-1,value='',pos=(120,70),size =
(100,20))
34         self.input_btn = wx.Button(panel,label = '确定',pos =
(240,70),size=(60,20))
35         self.inputText.Hide()
36         self.input_btn.Hide()
37         self.inputTxcl.Hide()
38
39         self.batchString = "这局游戏的黄金点数是"+str(self.golen_score)+"'\n
最近的玩家是'"+str(self.minId)+"'\n最远的玩家是'"+str(self.maxId)+"'"
40         self.batchText = wx.StaticText(panel,-1,label='',pos=(10,70),size=
(-1,-1))
41         self.batchText.Hide()
42
43         self.player_scoreString = ""
44         self.scoreText = wx.StaticText(panel,-1,label='',pos=(10,70),size=
(-1,-1))
45         self.scoreText.Hide()
46         ##### 绑定事件
47         self.strt_btn.Bind(wx.EVT_BUTTON,self.onClickStrtButton)
48         self.exit_btn.Bind(wx.EVT_BUTTON,self.onClickExitButton)
49         self.player_numTxcl.Bind(wx.EVT_TEXT,self.getInputPlayerNum)
50         self.pncm_btn.Bind(wx.EVT_BUTTON,self.onClickpncmButton)
51         self.input_btn.Bind(wx.EVT_BUTTON,self.onClickInputButton)
52         self.ctnu_btn.Bind(wx.EVT_BUTTON,self.onClickCtnuButton)
53         self.show_btn.Bind(wx.EVT_BUTTON,self.onClickShowButton)
54         self.Show()

```

“开始游戏”按钮被点击的事件处理函数

```

1     def onClickStrtButton(self,event):
2         self.strt_btn.Hide()
3         self.ctnu_btn.Show()
4         self.show_btn.Show()
5         self.exit_btn.Show()
6         self.player_numText.Show()
7         self.player_numTxcl.Show()
8         self.pncm_btn.Show()
9         # print("hello world")

```

“继续游戏”按钮被点击的事件处理函数

```

1     def onClickCtnuButton(self,event):
2         self.minId = self.maxId = -1
3         self.player_input = []
4         self.tmpId = 1
5         self.inputTxcl.SetValue('')
6         self.inputNotice = "请玩家" + str(self.tmpId) + "'输入"
7         self.inputText.SetLabel(self.inputNotice)
8         self.inputText.Show()
9         self.input_btn.Show()
10        self.inputTxcl.Show()
11        self.batchText.Hide()
12        self.scoreText.Hide()

```

“显示结果”

```

1     def onClickShowButton(self,event):
2         self.batchText.Hide()
3         for i,score in enumerate(self.player_score):
4             self.player_scoreString += "玩家"+str(i+1)+"'的分数
是'+str(score)+''\n"
5         self.scoreText.SetLabel(self.player_scoreString)
6         self.scoreText.Show()
7         self.player_scoreString = ""

```

“退出游戏”

```

1     def onClickExitButton(self,event):
2         print('exit')
3         self.Destroy()

```

输入参与游戏的总人数

```
1 def getInputPlayerNum(self,event):
2     self.player_num = int(self.player_numTxcl.GetValue())
3     # print(self.player_num)
4     pass
```

点击“确定”按钮，读取当前玩家输入

```
1 def onClickpncmButton(self,event):
2     self.player_numText.Hide()
3     self.player_numTxcl.Hide()
4     self.pncm_btn.Hide()
5     for i in range(self.player_num):
6         self.player_score.append(100)
7     self.inputText.Show()
8     self.input_btn.Show()
9     self.inputTxcl.Show()
```

读取玩家输入，若全部完成则显示游戏结果

```
1 def onClickInputButton(self,event):
2     self.player_input.append(int(self.inputTxcl.GetValue()))
3     print(self.player_input)
4     self.tmpId += 1
5
6     if (self.tmpId > self.player_num): #输入完了
7         self.golen_score = sum(self.player_input)/self.player_num*0.618
8         disList = [math.fabs(x-self.golen_score) for x in
self.player_input]
9         mx = -1.0
10        mn = 110.0
11        for i,x in enumerate(disList):
12            if x>mx:
13                self.maxId = i
14                mx = x
15            if x<mn:
16                self.minId = i
17                mn = x
18        self.player_score[self.minId] += self.player_num
19        self.player_score[self.maxId] -= 2
20        print(self.golen_score)
21        print(self.player_score)
22        self.inputTxcl.Hide()
23        self.inputText.Hide()
24        self.input_btn.Hide()
```

```

25         self.batchString = "这局游戏的黄金点数是'" + str(self.golen_score)
+ "'\n最近的玩家是'" + str(
26             self.minId+1) + "'\n最远的玩家是'" + str(self.maxId+1) + "'"
27         self.batchText.SetLabel(self.batchString)
28         self.batchText.Show()
29         return
30     self.inputTxcl.SetValue('')
31     self.inputNotice = "请玩家'" + str(self.tmpId) + "'输入"
32     self.inputText.SetLabel(self.inputNotice)

```

若为直接调用，则运行程序

```

1  if __name__ == '__main__':
2      app = wx.App()
3      frame = MyFrame()
4      app.MainLoop()

```

参考文档

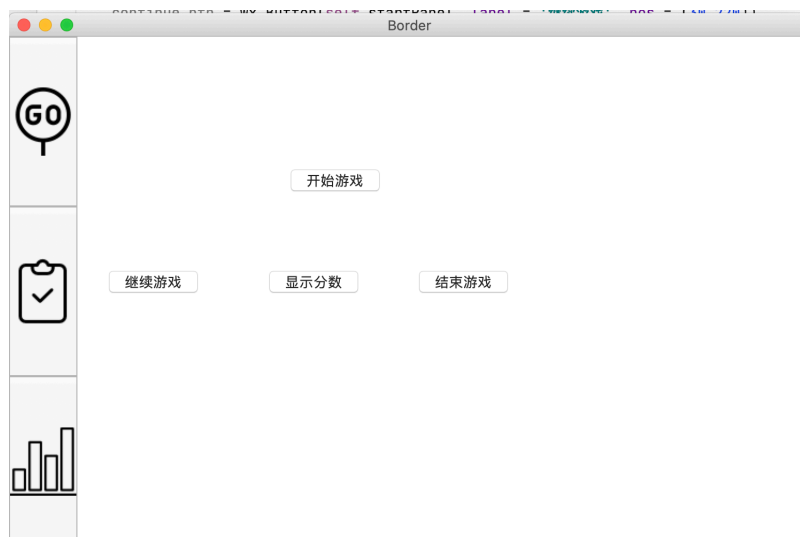
wxPython tutorial <http://zetcode.com/wxpython/>

week 11-12

工作进展

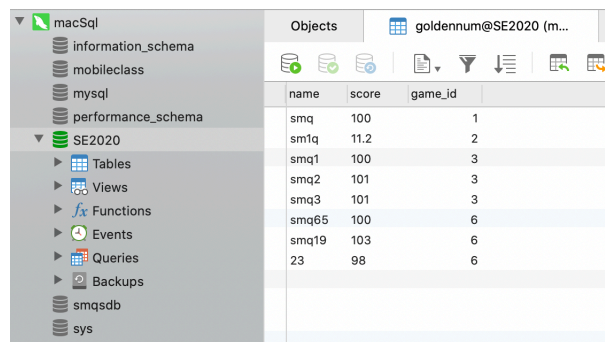
UI界面

对界面加入了图片元素，美化了UI界面。并且可以在不同的界面之中跳转，为实现更多的丰富和复杂的功能提供了准备。



数据库功能

实现了保存每局游戏的信息的功能。



The screenshot shows a MySQL database interface. On the left, a tree view displays the database structure, including 'macSql', 'information_schema', 'mobileclass', 'mysql', 'performance_schema', and 'SE2020'. The 'SE2020' database is selected, showing its contents: 'Tables', 'Views', 'Functions', 'Events', 'Queries', 'Backups', 'smqsdb', and 'sys'. On the right, the 'Objects' tab is active, displaying a table named 'smq' with the following data:

name	score	game_id
smq	100	1
sm1q	11.2	2
smq1	100	3
smq2	101	3
smq3	101	3
smq65	100	6
smq19	103	6
23	98	6

操作

实现了对键盘事件的监听，方便用户输入。

实现方法

UI界面

使用wx.BitmapButton给切换界面的按钮加上图片。

然后使用三个panel对应不同的面板。

使用wx.BoxSizer实现对控件大小、位置的控制。

```
1         mainBox = wx.BoxSizer(wx.HORIZONTAL)
2         mainBox.Add(sidebar)
3         mainBox.Add(mainPanel, wx.ID_ANY)
4         panel.SetSizer(mainBox)
5
6         sidebarBox = wx.BoxSizer(wx.VERTICAL)
7
8         startPanelButtonIcon = wx.Image('./static/img/start.png',
wx.BITMAP_TYPE_PNG).ConvertToBitmap()
9         self.startPanelButton = wx.BitmapButton(sidebar, wx.ID_ANY,
startPanelButtonIcon)
10        sidebarBox.Add(self.startPanelButton, wx.ALL)
11
12        resultPanelButtonIcon = wx.Image('./static/img/result.png',
wx.BITMAP_TYPE_PNG).ConvertToBitmap()
13        self.resultPanelButton = wx.BitmapButton(sidebar, wx.ID_ANY,
resultPanelButtonIcon)
14        sidebarBox.Add(self.resultPanelButton, wx.ALL)
15
16        graphPanelButtonIcon = wx.Image('./static/img/bar-chart.png',
wx.BITMAP_TYPE_PNG).ConvertToBitmap()
```

```
17         self.graphPanelButton = wx.BitmapButton(sidebar, wx.ID_ANY,
graphPanelButtonIcon)
18         sidebarBox.Add(self.graphPanelButton, wx.ALL)
19
20         sidebar.SetSizer(sidebarBox)
21
22         self.startPanel = wx.Panel(mainPanel, size = (700, 500))
23         self.startPanel.SetBackgroundColour('#ffffff')
24         self.resultPanel = wx.Panel(mainPanel, size = (700, 500))
25         self.resultPanel.SetBackgroundColour('#ffffff')
26         self.graphPanel = wx.Panel(mainPanel, size = (700, 500))
27         self.graphPanel.SetBackgroundColour('#ffffff')
28         self.resultPanel.Hide()
29         self.graphPanel.Hide()
30
31         start_btn = wx.Button(self.startPanel, label='开始游戏', pos = (200,
125))
32         continue_btn = wx.Button(self.startPanel, label = '继续游戏', pos =
(30,220))
33         show_btn = wx.Button(self.startPanel, label = '显示分数', pos =
(180,220))
34         exit_btn = wx.Button(self.startPanel, label = '结束游戏', pos =
(320,220))
35
36         self.startPanelButton.Bind(wx.EVT_BUTTON,
self.onStartPanelButtonClicked)
37         self.resultPanelButton.Bind(wx.EVT_BUTTON,
self.onresultPanelButtonClicked)
38         self.graphPanelButton.Bind(wx.EVT_BUTTON,
self.ongraphPanelButtonClicked)
39
40         def onStartPanelButtonClicked(self, event):
41             self.startPanel.Show()
42             self.resultPanel.Hide()
43             self.graphPanel.Hide()
44         def onresultPanelButtonClicked(self, event):
45             self.resultPanel.Show()
46             self.startPanel.Hide()
47             self.graphPanel.Hide()
48         def ongraphPanelButtonClicked(self, event):
49             self.graphPanel.Show()
50             self.startPanel.Hide()
51             self.resultPanel.Hide()
```


数据库保存

```
1  #数据库操作示例
2  import pymysql
3  conn = pymysql.connect(host = "localhost",user = "dataUser",password =
4  "scusmq61347",database = "SE2020",charset = "utf8")
5  cursor = conn.cursor()
6  sql = 'select count(*) from goldennum'
7  cursor.execute(sql)
8  print(cursor.fetchone())
9  #####
10     sql_insert = 'insert into goldennum values(%s,%s,%s)'
11     for i in range(self.player_num):
12         player = self.player_group[i]
13         args = [player.getName(),player.getScore(),self.game_id]
14         self.cursor.execute(sql_insert,args)
15     self.conn.commit()
16     self.conn.close()
17     print('exit')
```

键盘事件的监听

使用 `wx.EVT_TEXT_ENTER`

```
1  self.input_btn.Bind(wx.EVT_BUTTON or
2  wx.EVT_TEXT_ENTER,self.onClickInputButton)
```

下一阶段目标

- 实现网络请求的C/S模式、前后端通信
- 实现多个玩家远程游玩

week 13-14

工作进展

实现了用户登陆、注册功能

username ▾	password
asd	qwe
qwe	qwe
scu	qwe
smq	123456

注册

账号

密码

再次输入

失败信息
注册失败

实现了远程游玩的功能

- 已经部署到了服务器上
- 支持创建房间，可以让多个人一起远程游玩



实现方法

服务器端:

建立socket连接，为每个socket连接建立线程来处理通信

通过房间名和用户名来标识

并且可以保存每局游戏的结果

```

2  import socket
3  import threading
4  import pymysql
5  from Utils import sqlUtils, sendUtils
6  import math
7  userSocket = []
8  roomNum = {}
9  roomMax = {}
10 sendResult = {}
11 Result = {}
12 room_Score_dict = {}
13 room_Input_dict = {}
14 roomCreator = {}
15 room_result = {}
16 room_epoch = {}
17 room_epochCnt = {}
18 room_responseCnt = {}
19 room_Goldenscore = {}
20 roomname_gameid_dict = {}
21 max_gameid = -1
22
23 def s_sqlInit(host = 'localhost',user = 'dataUser',password =
    'scusmq61347',database = 'SE2020',charset = 'utf8'):
24     global conn,cursor
25     conn =
pymysql.connect(host=host,user=user,password=password,database=database,ch
arset=charset)
26     cursor = conn.cursor()
27
28 def s_sqlTest():
29     sql = 'select max(game_id) from goldennum'
30     cursor.execute(sql)
31     global max_gameid
32     max_gameid = int(cursor.fetchone()[0])
33     max_gameid+=1
34     print('max_gameid:',max_gameid)
35
36 def s_socketInit(host = 'localhost',port = 52345):
37     global sock
38     sock= socket.socket()
39     s_addr = (host, port)
40     sock.bind(s_addr)
41     sock.listen(60)
42
43 def conn_thread(soc):
44     username = ''
45     global
roomMax,roomMax,max_gameid,room_Score_dict,room_Input_dict,room_responseCn
t

```

```

46     while True:
47         data = soc.recv(1024)
48         jsData = eval(data)
49         operation = jsData['OPERATION']
50         if(operation=='login'):
51             args = (jsData['username'], jsData['password'])
52             sql = "select count(*) from userInfo where username = '%s'and
password='%s'"%args
53             cursor.execute(sql)
54             usernum = cursor.fetchone()[0]
55             if(usernum==1):
56                 username = jsData['username']
57                 sendUtils.s_loginSuccess(soc)
58             else:
59                 sendUtils.s_loginFailure(soc)
60
61         if(operation=='register'):
62             if(sqlUtils.uniqueUsername(conn, jsData['username'])):
63                 sqlUtils.insertUser(conn, jsData['username'],
jsData['password'])
64                 sendUtils.s_registerSuccess(soc)
65             else:
66                 sendUtils.s_registerFailure(soc)
67
68         if(operation=='username'):
69             sendUtils.s_sendUsername(soc,username)
70
71         if(operation=='enter'):
72             roomname = jsData['roomname']
73             if roomname not in roomNum:
74                 sendUtils.s_enterFailure(soc)
75             elif roomNum[roomname]>=roomMax[roomname]:
76                 sendUtils.s_enterFailure(soc)
77             else:
78                 sendUtils.s_enterSuccess(soc,max_gameid,roomNum[roomname])
79                 playerScore = room_Score_dict[roomname]
80                 roomNum[roomname] += 1
81                 playerScore[jsData['playername']] = 100
82
83         if(operation=='create'):
84             roomname = jsData['roomname']
85             maxnum = jsData['maxnum']
86             room_Score_dict[roomname] = {}
87             room_Input_dict[roomname] = {}
88             room_responseCnt[roomname] = 0
89             playerScore = room_Score_dict[roomname]
90             if roomname in roomMax:
91                 sendUtils.s_createFailure(soc)
92             else:

```

```

93         roomMax[roomname] = eval(maxnum)
94         roomNum[roomname] = 1
95         playerScore[jsData['playername']] = 100
96         room_epoch[roomname] = jsData['epoch']
97         room_epochCnt[roomname] = 0
98         roomCreator[roomname] = jsData['playername']
99         sendUtils.s_createSuccess(soc,max_gameid)
100         roomname_gameid_dict[roomname] = max_gameid
101         max_gameid += 1
102
103     if(operation=='ready'):
104         roomname = jsData['roomname']
105         tmpnum = roomNum[roomname]
106         tmpmax = roomMax[roomname]
107         if(tmpnum>=tmpmax):
108             sendUtils.s_readyOK(soc)
109         else:
110             sendUtils.s_readyNOTOK(soc)
111
112     if(operation=='input'):
113         roomname = jsData['roomname']
114         playername = jsData['playername']
115         input = eval(jsData['input'])
116         playerInput = room_Input_dict[roomname]
117         playerInput[playername] = input
118
119     if(operation=='result'):
120         roomname = jsData['roomname']
121         playername = jsData['playername']
122         playerInput = room_Input_dict[roomname]
123         playerscore = room_Score_dict[roomname]
124         input_num = len(playerInput)
125         player_num = roomMax[roomname]
126         creator = roomCreator[roomname]
127         far_name = ''
128         near_name = ''
129         dis_dict = {}
130         max_p = -1
131         min_p = 1000
132         golden_p = 0
133         result_str = ''
134         if creator==playername and input_num==player_num:
135             print('allInput:',playerInput)
136             for name,value in playerInput.items():
137                 golden_p+=value
138             golden_p /= player_num
139             golden_p *=0.618
140             for name,value in playerInput.items():
141                 dis_dict[name] = math.fabs(value-golden_p)

```

```

142         for name,dis in dis_dict.items():
143             if dis>max_p:
144                 max_p = dis
145                 far_name = name
146             if dis<min_p:
147                 min_p = dis
148                 near_name = name
149         playerscore[near_name] += player_num
150         playerscore[far_name] -= 2
151         for name,score in playerscore.items():
152             result_str += '玩家 '+name+'的分数为:'+str(score)+'\n'
153             result_str += '上次的winner是:' + near_name + '\n'
154             result_str += '上次的loser是:' + far_name + '\n'
155             result_str += '黄金点数是:' + str(golden_p) + '\n'
156
157         room_responseCnt[roomname] = player_num
158         room_result[roomname] = result_str
159         room_epochCnt[roomname] += 1
160
161
162         if room_responseCnt[roomname]>0:
163             result_str = room_result[roomname]
164             end = room_epochCnt[roomname]==room_epoch[roomname]
165             ok = 'OK'
166
167         sendUtils.s_result(soc,result_str,end,ok,room_Input_dict[roomname])
168         room_responseCnt[roomname] -= 1
169         if room_responseCnt[roomname]==0:
170             del room_Input_dict[roomname]
171             room_Input_dict[roomname] = {}
172             if room_epochCnt[roomname]==room_epoch[roomname]:
173
174                 sqlUtils.insertHistory(conn,playername,playerscore[playername],roomname_g
175 ameid_dict[roomname])
176             else:
177                 sendUtils.s_result(soc,None,False,'NOTOK',None)
178
179 s_socketInit()
180 s_sqlInit()
181 s_sqlTest()
182
183 while True:
184     c,addr = sock.accept()
185     print('连接地址: '+str(addr))
186     # Send.s_loginSuccess(c)
187     userSocket.append(c)
188     handle_thread = threading.Thread(target=conn_thread,args=(c,))
189     handle_thread.start()
190     print('ok')

```

客户端:

client.py

作为客户端入口，提供self.updateFrame回调函数，实现面板的切换

```

1  import wx
2  import socket
3  from Utils import guiManager as FrameManager
4
5  sock = socket.socket()
6  # host = '47.106.229.249'
7  host = 'localhost'
8  port = 52345
9  addr = (host,port)
10 sock.connect(addr)
11
12 def jsonParse(js):
13     recjs = eval(js)
14     if recjs['MESSAGE'] == 'SUCCESS':
15         return True
16
17 data = ''
18
19 def recvMsg():
20     while True:
21         try:
22             data = sock.recv(1024)
23         except:
24             continue
25         else:
26             jsonParse(data)
27             # time.sleep(0.5)
28             # print('in thread')
29
30 class clientApp(wx.App):
31     def OnInit(self):
32         self.manager = FrameManager.guiManager(self.updateFrame,sock)
33         # self.frame = loginFrame.LoginFrame(sock)
34         self.frame = self.manager.getFrame(0)
35         self.SetTopWindow(self.frame)
36         self.frame.Show(True)
37         return True
38     def OnExit(self):
39         sock.close()
40

```



```

41     def updateFrame(self, type, roomname=None, gameid=None, playername=None):
42         self.frame.Show(False)
43         self.frame = self.manager.getFrame(type, roomname, gameid, playername)
44         self.SetTopWindow(self.frame)
45         self.frame.Show(True)
46
47
48 if __name__ == "__main__":
49     # recvThread = threading.Thread(target=recvMsg, args=())
50     # recvThread.start()
51     app = clientApp()
52     app.MainLoop()
53

```

loginFrame.py

实现登陆界面，用wx.BoxSizer管理控件

```

1  import wx
2  from Utils import sendUtils
3  from frames import registerDialog
4
5
6  class LoginFrame(wx.Frame):
7      def __init__(self, sock, parent=None, id=-1, updateFrame = None):
8          super(LoginFrame, self).__init__(parent=None, title = '黄金点数游
9          戏', size = (400, 350))
10         self.panel = wx.Panel(self, -1)
11         self.sock = sock
12         self.updateFrame = updateFrame
13
14         # v_box_sizer = wx.BoxSizer(wx.VERTICAL)
15
16         self.bt_confirm = wx.Button(self.panel, label="登陆") # 创建按钮
17         self.bt_confirm.Bind(wx.EVT_BUTTON, self.OnclickSubmit)
18         self.bt_cancel = wx.Button(self.panel, label="注册")
19         self.bt_cancel.Bind(wx.EVT_BUTTON, self.OnclickRegister)
20
21         self.title = wx.StaticText(self.panel, label="登陆")
22
23         self.label_user = wx.StaticText(self.panel, label="用户名:")
24         self.text_user = wx.TextCtrl(self.panel, style=wx.TE_LEFT)
25
26         self.label_pwd = wx.StaticText(self.panel, label="密 码:")
27         self.text_password = wx.TextCtrl(self.panel, style=wx.TE_PASSWORD)
28         # 控件横向排列
29         hsizer_user = wx.BoxSizer(wx.HORIZONTAL)
30         hsizer_user.Add(self.label_user, proportion=0, flag=wx.ALL,
31         border=5)

```

```

30         hsize_user.Add(self.text_user, proportion=1, flag=wx.ALL,
border=5) # proportion=0表示不变,proportion=1两倍宽度
31
32         hsize_pwd = wx.BoxSizer(wx.HORIZONTAL)
33         hsize_pwd.Add(self.label_pwd, proportion=0, flag=wx.ALL,
border=5)
34         hsize_pwd.Add(self.text_password, proportion=1, flag=wx.ALL,
border=5)
35
36         hsize_button = wx.BoxSizer(wx.HORIZONTAL)
37         hsize_button.Add(self.bt_confirm, proportion=0,
flag=wx.ALIGN_CENTER, border=5)
38         hsize_button.Add(self.bt_cancel, proportion=1,
flag=wx.ALIGN_CENTER, border=5)
39         # 控件纵向排列
40         vsizer_all = wx.BoxSizer(wx.VERTICAL)
41         vsizer_all.Add(self.title, proportion=0, flag=wx.BOTTOM | wx.TOP |
wx.ALIGN_CENTER, border=15)
42
43         vsizer_all.Add(hsize_user, proportion=0, flag=wx.EXPAND | wx.LEFT
| wx.RIGHT, border=45)
44         vsizer_all.Add(hsize_pwd, proportion=0, flag=wx.EXPAND | wx.LEFT
| wx.RIGHT, border=45)
45         vsizer_all.Add(hsize_button, proportion=0, flag=wx.ALIGN_CENTER,
border=15)
46         self.panel.SetSizer(vsizer_all)
47
48         font = wx.Font(16, wx.DEFAULT, wx.FONTSTYLE_NORMAL, wx.NORMAL,
underline=False)
49         self.title.SetFont(font)
50
51
52         def validUser(self,soc):
53             data = soc.recv(1024)
54             jsdata = eval(data)
55             print('jsdata'+str(jsdata))
56             if(jsdata['MESSAGE']=='success'):
57                 return True
58             else:
59                 return False
60         def valiRegister(self,soc):
61             data = soc.recv(1024)
62             jsdata = eval(data)
63             if(jsdata['MESSAGE']=='success'):
64                 return True
65             else:
66                 return False
67
68         def OnclickSubmit(self, event):

```

```

69         self.username = self.text_user.GetValue()
70         self.password = self.text_password.GetValue()
71         sendUtils.c_sendLogInfo(self.sock, self.username, self.password)
72
73         if(self.validUser(self.sock)):
74             # self.panel.Parent.Hide()
75             # app = roomFrame.MainApp()
76             # app.MainLoop()
77             self.updateFrame(1)
78         else:
79             message = '登陆失败, 用户名或密码错误'
80             wx.MessageBox(message, '登陆失败')
81
82         def OnclickRegister(self, event): #注册
83             dlg = RegisterWindow(self.getRegisterInfo, '#0a74f7')
84             dlg.Show()
85
86         def getRegisterInfo(self, username, password):
87             self.r_username = username
88             self.r_password = password
89             sendUtils.c_sendRegisterInfo(self.sock, self.r_username,
self.r_password)
90             print('callback'+username)
91             if(self.valiRegister(self.sock)):
92                 return
93             else:
94                 wx.MessageBox('注册失败', '失败信息')
95
96
97     class RegisterWindow(registerDialog.RegisterDialog):
98         def __init__(self, callback_func, themeColor):
99             registerDialog.RegisterDialog.__init__(self, '注册',
callback_func, themeColor)
100

```

registerDialog.py

实现注册，往数据库中写入数据，并且支持注册检测

```

1  import wx
2
3
4  class RegisterDialog(wx.Dialog):
5      def __init__(self, title, func_callBack, themeColor):
6          wx.Dialog.__init__(self, None, -1, title, size=(300, 200))
7          self.func_callBack = func_callBack
8          self.themeColor = themeColor
9
10         self.InitUI() # 绘制Dialog的界面

```

```

11
12     def InitUI(self):
13         panel = wx.Panel(self)
14
15         font = wx.Font(14, wx.DEFAULT, wx.BOLD, wx.NORMAL, True)
16
17         accountLabel = wx.StaticText(panel, -1, '账号', pos=(20, 25))
18         accountLabel.SetFont(font)
19
20         self.accountInput = wx.TextCtrl(panel, -1, u'', pos=(80, 25), size=
(180, -1))
21         self.accountInput.SetForegroundColour('gray')
22         self.accountInput.SetFont(font)
23
24         passwordLabel1 = wx.StaticText(panel, -1, '密码', pos=(20, 70))
25         passwordLabel1.SetFont(font)
26         self.passwordInput1 = wx.TextCtrl(panel, -1, u'', pos=(80, 70),
size=(180, -1), style=wx.TE_PASSWORD)
27         self.passwordInput1.SetFont(font)
28
29         passwordLabel2 = wx.StaticText(panel, -1, '再次输入', pos=(20, 100))
30         passwordLabel2.SetFont(font)
31         self.passwordInput2 = wx.TextCtrl(panel, -1, u'', pos=(80, 100),
size=(180, -1), style=wx.TE_PASSWORD)
32         self.passwordInput2.SetFont(font)
33
34         sureButton = wx.Button(panel, -1, u'注册', pos=(20, 130), size=
(120, 40))
35         self.Bind(wx.EVT_BUTTON, self.sureEvent, sureButton)
36
37         cangleButton = wx.Button(panel, -1, u'取消', pos=(160, 130), size=
(120, 40))
38
39         # 为【取消Button】绑定事件
40         self.Bind(wx.EVT_BUTTON, self.cangleEvent, cangleButton)
41
42     def sureEvent(self, event):
43         account = self.accountInput.GetValue()
44         password1 = self.passwordInput1.GetValue()
45         password2 = self.passwordInput2.GetValue()
46         if password1!=password2:
47             wx.MessageBox('两次输入的密码不一致', '注册失败')
48             return
49         # 通过回调函数传递数值
50         self.func_callBack(account, password1)
51         self.Destroy() # 销毁隐藏Dialog
52
53     def cangleEvent(self, event):
54         self.Destroy() # 销毁隐藏Dialog

```

roomFrame.py

通过房间名创建、加入房间

```
1  import wx
2  from Utils import sendUtils
3
4
5  class RoomFrame(wx.Frame):
6      def __init__(self, sock,parent=None,id=-1,updateFrame = None):
7          wx.Frame.__init__(self, parent, -1, title="游戏房间", size=(400,
350),
8
9                      style=wx.DEFAULT_FRAME_STYLE & ~(wx.RESIZE_BORDER
10 | wx.MAXIMIZE_BOX))
11
12          self.panel = wx.Panel(self,-1)
13          self.sock = sock
14          self.updateFrame = updateFrame
15          self.username = self.getUsername(self.sock)
16          self.gameid = 0
17
18
19          print('self.username:',self.username)
20          self.initUI()
21
22      def initUI(self):
23          self.bt_create = wx.Button(self.panel,label="创建房间")
24          self.bt_create.Bind(wx.EVT_BUTTON, self.OnclickCreate)
25          self.bt_enter = wx.Button(self.panel,label="加入房间")
26          self.bt_enter.Bind(wx.EVT_BUTTON,self.OnclickEnter)
27          hsizer_control = wx.BoxSizer(wx.HORIZONTAL)
28
29          hsizer_control.Add(self.bt_create,proportion=0,flag=wx.ALL,border=5)
30          hsizer_control.Add(self.bt_enter,proportion=0,flag=wx.ALL,border=5)
31
32          self.label_name = wx.StaticText(self.panel,label="房间名")
33          self.text_name = wx.TextCtrl(self.panel,style=wx.TE_LEFT)
34          self.label_num = wx.StaticText(self.panel,label="房间人数")
35          self.text_num = wx.TextCtrl(self.panel,style=wx.TE_LEFT)
36          hsizer_input = wx.BoxSizer(wx.HORIZONTAL)
37          hsizer_input.Add(self.label_name,proportion=0,flag=wx.ALL,border=5)
38          hsizer_input.Add(self.text_name, proportion=1, flag=wx.ALL,
border=5)
39          hsizer_input.Add(self.label_num, proportion=0, flag=wx.ALL,
border=5)
40          hsizer_input.Add(self.text_num, proportion=1, flag=wx.ALL,
border=5)
41
42          self.title = wx.StaticText(self.panel, label="游戏大厅")
```

```

39         font = wx.Font(16, wx.DEFAULT, wx.FONTSTYLE_NORMAL, wx.NORMAL,
underline=False)
40         self.title.SetFont(font)
41         vsizer_all = wx.BoxSizer(wx.VERTICAL)
42         vsizer_all.Add(self.title, proportion=0, flag=wx.BOTTOM | wx.TOP |
wx.ALIGN_CENTER, border=15)
43         vsizer_all.Add(hsizer_input, proportion=0, flag=wx.EXPAND | wx.LEFT
| wx.RIGHT, border=45)
44         vsizer_all.Add(hsizer_control, proportion=0, flag=wx.ALIGN_CENTER,
border=15)
45         self.panel.SetSizer(vsizer_all)
46
47     def OnclickCreate(self, event):
48         roomname = self.text_name.GetValue()
49         maxnum = self.text_num.GetValue()
50         sendUtils.c_createRoom(self.sock, roomname, maxnum, self.username)
51         if self.validCreate(self.sock):
52             self.updateFrame(2, roomname, self.gameid, self.username)
#playerid == 0
53         else:
54             wx.MessageBox('创建房间失败, 请更换房间名', '创建失败')
55     def OnclickEnter(self, event):
56         roomname = self.text_name.GetValue()
57         sendUtils.c_enterRoom(self.sock, roomname, self.username)
58         if self.validEnter(self.sock):
59             self.updateFrame(2, roomname, self.gameid, self.username)
60         else:
61             wx.MessageBox('加入房间失败 (房间不存在或人数已满)', '加入失败')
62
63     def validCreate(self, soc):
64         data = soc.recv(1024)
65         jsdata = eval(data)
66         if(jsdata['MESSAGE']=='success'):
67             self.gameid = jsdata['gameid']
68             return True
69         else:
70             return False
71
72     def validEnter(self, soc):
73         data = soc.recv(1024)
74         jsdata = eval(data)
75         if(jsdata['MESSAGE']=='success'):
76             self.gameid = jsdata['gameid']
77             self.playerid = jsdata['playerid']
78             return True
79         else:
80             return False
81
82     def getUsername(self, soc):

```

```

83         sendUtils.c_getUsername(soc)
84         data = soc.recv(1024)
85         jsdata = eval(data)
86         return jsdata['MESSAGE']
87
88     class MainApp(wx.App):
89         def OnInit(self):
90             self.frame = RoomFrame(None)
91             self.frame.Show()
92             return True
93
94     if __name__ == "__main__":
95         app = MainApp()
96         app.MainLoop()
97

```

gameFrame.py

通过线程来等待其他玩家的输入和最终的结果

```

1  import wx
2  import math
3  import pymysql
4  import threading
5  from Utils import sendUtils
6  import time
7  from threading import Timer
8  class GameFrame(wx.Frame):
9      def
10         __init__(self, sock, roomname=None, gameid=None, playername=None, parent=None, i
d=-1, updateFrame=None):
11             wx.Frame.__init__(self, parent=None, title='黄金点数游
戏,'+roomname, size = (450,350), style=wx.DEFAULT_FRAME_STYLE & ~
(wx.RESIZE_BORDER | wx.MAXIMIZE_BOX))
12
13             self.sock = sock
14             self.updateFrame = updateFrame
15             self.gameid = gameid
16             self.playername = playername
17             print('game creator:', self.playername)
18             self.roomname = roomname
19             print('playerid:', playername)
20             waiting_thread = threading.Thread(target=self.waiting, args=
(self.sock,))
21             waiting_thread.start()
22
23             self.initWidgets()
24             self.bindEvents()

```

```

25
26         self.Show()
27
28     def initWidgets(self):
29         self.panel = wx.Panel(self)
30         self.label_waiting = wx.StaticText(self.panel, -1, label = '请等待其
他玩家加入', pos=(10, 70))
31         self.label_waittothers = wx.StaticText(self.panel, -1, label='请等待
其他玩家输入', pos=(10, 70))
32         self.label_waittothers.Hide()
33
34         self.label_input = wx.StaticText(self.panel, -1, label = '请输入您的点
数', pos=(10, 70))
35         self.text_input = wx.TextCtrl(self.panel, -1, value='', pos=(120, 70))
36         self.bt_input = wx.Button(self.panel, label='确定', pos =
(240, 70), size=(60, 20))
37         self.hideInput()
38
39         self.text_result = wx.StaticText(self.panel, -1, label='', pos=
(10, 120))
40         self.text_wait = wx.StaticText(self.panel, -1, label='五秒后回到输入')
41         self.hideResult()
42
43         self.bt_return = wx.Button(self.panel, label='回到房间', pos =
(240, 70), size=(100, 20))
44         self.bt_return.Bind(wx.EVT_BUTTON, self.OnclickReturn)
45         self.bt_return.Hide()
46
47     def OnclickInput(self, event):
48         point = self.text_input.GetValue()
49         sendUtils.c_point(self.sock, point, self.playername, self.roomname)
50         self.hideInput()
51         self.label_waittothers.Show()
52         result_thread = threading.Thread(target=self.getResult, args=
(self.sock,))
53         result_thread.start()
54
55     def OnclickReturn(self, event):
56         self.updateFrame(1)
57     def bindEvents(self):
58         self.bt_input.Bind(wx.EVT_BUTTON, self.OnclickInput)
59
60     def showResult(self):
61         self.text_result.Show()
62         self.text_wait.Show()
63     def hideResult(self):
64         self.text_result.Hide()
65         self.text_wait.Hide()
66

```



```

67     def showInput(self):
68         self.label_input.Show()
69         self.text_input.Show()
70         self.bt_input.Show()
71
72     def hideInput(self):
73         self.label_input.Hide()
74         self.text_input.Hide()
75         self.bt_input.Hide()
76
77     def initData(self):
78         self.player_num = 0
79         self.golden_score = 0
80         self.player_group = []
81         self.player_input = []
82         self.tmpId = 0
83         self.maxId = -1
84         self.minId = -1
85         # self.player_group.append(Player(str(0)))
86         self.conn = pymysql.connect(
87             host = "localhost",
88             user = "dataUser",
89             password = "scusmq61347",
90             database = "SE2020",
91             charset = "utf8"
92         )
93         sql_count = 'select count(*) from goldennum'
94         self.cursor = self.conn.cursor()
95         self.cursor.execute(sql_count)
96         self.game_id = self.cursor.fetchone()[0]+1
97         self.firstGame = True
98
99     def waiting(self,soc):
100         while True:
101             time.sleep(0.05)
102             sendUtils.c_readyQuery(soc,self.roomname)
103             data = soc.recv(1024)
104             jsdata = eval(data)
105             if(jsdata['MESSAGE']=='OK'):
106                 self.ready = True
107                 print('ready ok')
108                 self.label_waiting.Hide()
109                 self.showInput()
110                 return
111
112     def showInputAgain(self):
113         self.showInput()
114         self.hideResult()
115

```

```
116     def getResult(self,soc):
117         while True:
118             time.sleep(0.05)
119             sendUtils.c_resultQuery(soc,self.playername,self.roomname)
120             data = soc.recv(1024)
121             jsdata = eval(data)
122             if(jsdata['MESSAGE']=='OK'):
123                 self.label_waitothers.Hide()
124                 self.showResult()
125                 self.text_result.SetLabel(jsdata['result'])
126                 self.all_input = jsdata['all_input']
127                 print(self.all_input)
128                 end = jsdata['end']
129                 if not end:
130                     t = Timer(5.0,self.showInputAgain)
131                     t.start()
132                 else:
133                     self.bt_return.Show()
134                     self.text_wait.SetLabel('游戏结束')
135                 return
136
137 if __name__ == '__main__':
138     app = wx.App()
139     frame = GameFrame(None)
140     app.MainLoop()
141
```