دانشگاه صنعتی شریف

نحوه‌ی استفاده از رابط سرور - کلاینت

درس برنامه سازی پیشرفته

کامیار میرزازاد

بهار ۹۳

- **نحوه‌ی استفاده از API کلاینت**

API کلاینت از دو ابژکت کد client.o ، client_proxy.o و هدر client_proxy.h تشکیل می‌شود که برای استفاده از آنها در محیط QT نخست لازم است آنها را به پروژه خود اضافه کنید. برای این منظور نخست لازم است این سه فایل در پوشه اصلی پروژه خود (پوشه حاوی سورس کدها) قرار داده و سپس تنظیمات زیر را به انتهای محتویات فایل pro. پروژه خود **اضافه** کنید :

```
QT       += network

HEADERS += \
   client_proxy.h

OBJECTS += \
   client_proxy.o \
   client.o
```

- ممکن است پس از اعمال تغییرات بالا فرآیند build پروژه QT دچار مشکل شود که در این صورت بستن و دوباره بازکردن پروژه این مشکل را برطرف خواهد کرد.

پس از اضافه کردن خطهای بالا به نتظیمات پروژه QT خود می توانید در main.cpp خود از هدر client_proxy.h استفاده کنید :

```
#include “client_proxy.h”
```

این هدر شامل توابعی است که به شما اجازه می‌دهد با سرور ارتباط برقرار کنید. در ادامه این توابع را تک تک توضیح می‌دهیم :

- **`client_proxy( std::string username )`**

این تابع سازنده کلاس client_proxy بوده که قبل از هرچیزی لازم است یک نسخه از آن بسازیم. این تابع به عنوان ورودی یک string می‌گیرد که نام کاربر است.

- **`bool login( QString serverIP )`**

این تابع ارتباط کلاینت با سرور را برقرار می‌کند : با فراخوانی این API تابع کلاینت سعی خواهد کرد که به سرور با آدرس IP داده شده وصل شود.

- درصورتی بازی هنوز شروع نشده باشد و تعداد کاربرهای متصل سرور به حداکثر نرسیده باشد، مقدار بازگشتی این تابع true خواهد بود و return کردن این تابع به معنی شروع بازی خواهد بود.
- چنانچه تعداد کاربرها به حداکثر رسیده باشد، تابع مقدار false را برمی‌گرداند.
- درصورتی بازی شروع شده باشد، با توجه به اینکه سرور دیگر منتظر اتصال کلاینت‌ها نیست ، این تایع پس از ۲ دقیقه timeout داده و برنامه را خاتمه می‌دهد.

همانطور که در بالا ذکر شد برای استفاده از این تابع لازم است شما آدرس IP سرور را در قالب یک رشته به آن پاس کنید. آدرس IP درصورتی سرور با کلاینت در یک ماشین درحال اجرا باشد "127.0.0.1" خواهد بود که همانطور که احتمالا در درس شبکه دیده‌اید به معنی loopback هست. در این حالت لزومی نیست که ماشین شما به اینترنت یا شبکه دیگری متصل باشد. در صورتی بخواهید سرور خود را برروی ماشین دیگری را اجرا کنید، لازم است که آدرس IP آن را به تابع بالا پاس کنید.

- آدرس IP ماشین خود را در سیستم عامل linux می‌توانید با وارد کردن دستور ifconfig در ترمینال مشاهده کنید. متناظر این دستور در windows ، ipconfig است.

آدرس IP سروری که برای مسابقه نهایی از آن استفاده خواهد شد، متعاقبا به شما اعلام خواهد شد.

- **`void logout()`**

با فراخوانی این تابع ارتباط کلاینت با سرور قطع خواهد شد. کاربرد درنظر گرفته شده برای این تابع فرخوانی آن پس از پایان بازی است . در صورتی که این تابع را قبل از اتمام بازی فراخوانی کنید، دسترسی خود به عامل‌ها و داشنکده‌هایتان را از دست خواهید داد.

- **`int get_my_id()`**

این تابع id درنظرگرفته شده برای کلاینت شما توسط سرور را برخواهد گرداند. این رقم عددی بین ۱ و بیشینه تعداد کاربرها خواهد بود.

- **`int get_winner_id()`**

فراخوانی این تابع پس از پایان بازی id بازیکن برنده را برخواهد گرداند. درصورتی که قیل از فراخوانی بازی این تابع را فراخوانی کنید، مقدار برگردانده شده *تصادفی* خواهد بود.

- **`bool wait_for_tick ()`**

فراخوانی این تابع به معنی اتمام حرکات کلاینت شما در دور جاری بازی خواهد بود. اجرای این تابع درصورتی اتمام می‌یابد که سرور دور بعدی بازی را شروع کند. مقدار برگردانده شده توسط این تابع درصورت اتمام بازی در دور جاری true و در غیر این صورت false خواهد بود. با توجه به اینکه سرور در انتهای هردور نقشه بازی را به روزرسانی می‌کند ، درصورتی که بخواهید نقشه جدید را از سرور دریافت کنید نخست لازم است با فراخوانی تابع بالا از اتمام این دور اطمینان حاصل کنید.

توجه داشته باشید که زمان درنظر گرفته شده برای هردور بازی مشخص بوده و درصورتی کلاینت شما در طول این بازه تابع فوق را فراخوانی نکند، سرور منتظر پایان حرکات شما نخواهد بود و دور بعدی را شروع خواهد کرد. چنانچه کلاینت شما فراخوانی این تابع در یک دور از بازی را فراموش کند ، فراخوانی بعدی برای دور بعدی خواهد بود نه برای دور سپری شده

- **`bool move_agent ( int agent_id , std::string direction )`**

همانطور که از اسم آن نیز مشخص است ، با فراخوانی این تابع سرور عامل با شناسه agent_id شما را در جهت مشخص شده جابجا خواهد کرد. درصورتی که سرور در این کار موفق شود، این تابع مقدار true برخواهد گرداند ولی چنانچه شما عاملی با شناسه agent_id نداشته باشید یا اینکه نوبت حرکت آن نباشد یا اینکه عامل نتواند در جهت مشخص شده حرکت کند ، این مقدار false خواهد بود. رشته‌های حرفی درنظر گرفته شده برای ۸ جهت در زیر آمده است :

- U,L,R,D,UL,UR,DL,DR

عملکرد تابع به ازای رشته‌های حرفی به به غیر از موارد ذکر شده در بالا *تصادفی* خواهد بود.

- **bool change_department_product(int building_id , char desired_type)**

همانطور که از اسم آن نیز مشخص است ، با فراخوانی این تابع سرور نوع عامل تولیدی توسط دانشکده با شناسه department_id را به نوع مشخص شده تبدیل خواهد کرد. درصورتی که سرور در این کار موفق شود، این تابع مقدار true برخواهد گرداند ولی چنانچه شما صاحب دانشکده با شناسه department_id نباشید، این مقدار false خواهد بود. کاراکترهای درنظر گرفته شده برای ۳ نوع عامل در زیر آمده است :

- S,T,P

عملکرد تابع به ازای کاراکترهایی به غیر از موارد ذکر شده در بالا *تصادفی* خواهد بود.

- **void get_map (char* map)**

پس از فراخوانی این تابع نقشه فعلی بازی در محلی که اشاره‌گر پاس شده مشخص می‌کند، کپی خواهد شد.

- **void print_map ()**

فراخوانی این تابع نقشه بازی را به یک فرمت ساده چاپ خواهد کرد.

- درصورتی که این تابع را در کد خود استفاده می‌کنید، بهتر است برنامه خود را ، که در داخل پوشه build قرار دارد ، به صورت مستقیم *(بدون وساطت QT)* از ترمینال اجرا کنید.

- **bool execute (std::string command)**

این تابع صرفا یک واسطه برای اجرای دو دستور **move_agent** و **change_department_product** بوده و دستورهای به فرمت زیر را فراخوانی کرده و خروجی آنها را برمی‌گرداند:

- move [agent_id] [direction]
- change [building_id] [type]

برای درک بهتر نحوه‌ی عملکرد API کلاینت می‌توانید کارکرد پروژه QT نمونه‌ای را که با استفاده از توابع بالا نوشته شده است، بررسی کنید.

- **نحوه‌ی استفاده از سرور**

برای اجرا فایل اجرای سرور نخست لازم است پلتفرم ژاوا را برروی کامپیوتر خود نصب کنید. البته این پلتفرم عمدتا به صورت پیش فرض در سیستم عمل لینوکس موجود است. به هرحال با استفاده از دستور زیر می‌توانید نسخه ۱,۷این پلتفرم را نصب کنید :

```
sudo apt-get install openjdk-7-jre
```

پس از اتمام عمل نصب ، محتویات فایل فشرده شده AP_Server_Executable_and_Config.zip را در یک پوشه دلخواه قرار داده ، سپس در ترمینال با استفاده از دستور cd [directory] وارد آن پوشه شوید. حال با وارد کردن دستور زیر در ترمینال سرور را اجرا کنید :

```
java –jar AP_Server.jar
```

توجه داشته باشید که در حین راه‌اندازی سرور فایل server.txt که حاوی نتظیمات سرور است، لازم در همان پوشه قرار داشته باشد. این فایل یک فایل متنی است که

- در خط اول آن نخست بیشینه تعداد دورهای بازی و سپس زمان درنظر گرفته شده برای هر دور (به میلی ثانیه) آمده است.
- در خط دوم نخست طول و عرض نقشه بازی سپس بیشینه تعداد کاربرهای سرور و در پایان تعداد دورهای بین تولید دو عامل داعش مشخص شده است.
- هریک از خط‌های باقی‌مانده ، درصورت وجود، مشخصات دانشکده ها را مشخص می‌کنند. ترتیب این جزئیات به صورت زیر است :

شناسه دانشکده – مختصات گوشه‌ی بالا سمت چپ – مختصات در ورودی دانشکده – طول و عرض دانشکده – جان (health) دانشکده – بازه‌ی زمانی بین تولید دو عامل توسط آن دانشکده – هزینه upgrade کردن دانشکده (که از power آن کم می‌کند)

سرور پس از شروع به کار تا زمانی که کلید enter را فشار دهید ، منتظر کلاینت‌های جدید خواهد بود لذا برای شروع بازی کافی است کلید enter را فشار دهید.

- **تغییرات نقشه**

با توجه به نیازهای نسخه شبکه‌ای بازی مجبور شدیم فیلدهای زیر را به نقشه بازی اضافه کنیم :

- فیلد health که با یک فاصله به آخر لیست فیلدهای agent اضافه شده است.
- سه فیلد power ، level (درجه upgrade دانشکده) و health که به ترتیب با یک فاصله به انتهای لیست فیلدهای department اضافه شدند.