به نام خداوند بخشنده‌ی مهربان

تمرین برنامه‌نویسی صفرم[1]

شبکه‌های کامپیوتری

مدرس: مهدی خرازی **پاییز ۱۳۹۱**

در این تمرین شما باید یک بازی را به صورت P2P با استفاده از مفاهیم Socket در محیط Unix به زبان C یا C++ و با کمک کتابخانه‌ی OpenGL پیاده‌سازی کنید.

هدف

آشنایی با

- محیط سامانه‌های عامل UNIX
- محیط برنامه‌نویسی Eclipse
- UNIX Socket API
- POSIX Thread API
- OpenGL API
- نوشتن Makefile
- ساختن پوشه، نام‌گذاری و فشرده‌سازی و ارسال صحیح تمرین

مقدمه

برای طراحی یک بازی روی شبکه تمهیدات خاصی باید اندیشیده شود. اتصال بین بازیکن‌ها یکی از مهمترین موارد است. راه‌کار ساده‌ی برقراری این ارتباط، استفاده از یک کارگزار برای ساماندهی ارتباط بین بازیکن‌های مختلف است. در این روش در صورتی که کارگزار در دسترس نباشد، امکان ارتباط بین بازیکن‌ها وجود نخواهد داشت. در این تمرین قصد داریم یک بازی تحت شبکه به سبک Tower Defense را پیاده‌سازی کنیم که نیازی به یک کارگزار مرکزی نداشته باشد.

[1] با تشکر از بهنام مومنی، مهدی احمدی‌نژاد، کامیار اللهوردی، سجاد فولادی، علی‌محمد ربانی، روزبه کتابی و رامتین رطبی

برنامه‌ی بازیکن

شما تنها باید این برنامه را پیاده‌سازی کنید. برنامه‌ی بازیکن برای اجرا نیاز به ورودی‌های زیر دارد:

$ ./player [listen-port] [next-ip] [next-port] [map-file] [player-index] [starter?]

برای مثال:

$ ./player 8080 127.0.0.1 8081 map.txt 0 true

ورودی‌ها در جدول زیر شرح داده شده‌اند:

listen-port	**پورتی که باید بازیکن قبلی با آن به این بازیکن وصل شود**
next-ip	**آدرس بازیکن بعدی**
next-port	**پورت بازیکن بعدی**
map-file	**آدرس فایل حاوی نقشه‌ی بازی**
player-index	**اندیس بازیکن در لیست بازیکن‌های موجود در فایل نقشه**
starter?	**آیا این بازیکن شروع کننده‌ی بازی است؟** (مقدار **true** یا **false**)

فایل حاوی نقشه به صورت زیر ساخته می‌شود:

```
<players_count>
<player1-name> <player1-type>
…
<rows> <cols>
<celltype1-1> <celltype1-2>… <celltype1-<cols>>
<celltype2-1> <celltype2-2>… <celltype2-<cols>>
…
<celltype<rows>-1> <celltype<rows>-2>… <celltype<rows>-<cols>>
```

مقدارهای ممکن <playeri-type> در جدول زیر آمده است:

Team Type	Value
Soldier Team	soldier
Tower Team	tower

مقدارهای <celltype-i-j> یکی از حالت‌های زیر می‌تواند باشد:

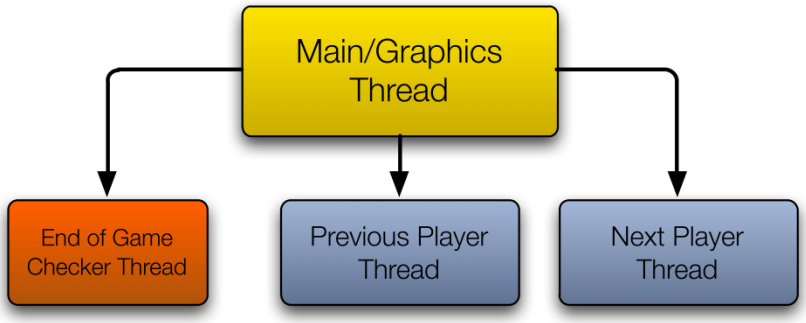
Type	Celltype Value
Start	S
End	E
Forest	-
Path	*

برای نمونه در زیر یک نمونه از فایل نقشه آمده است:

```
2
DarkWarriors soldier
HolyTowers tower
10 10
- - - - - - - - - -
S * * * * * * * * -
- - - - - - - - * -
- * * * * * * * * -
- * - - - - - - - -
- * * * * * * * * -
- - - - - - - - * -
- * * * * * * * * -
- * - - - - - - - -
- E - - - - - - - -
```

جریان بازی

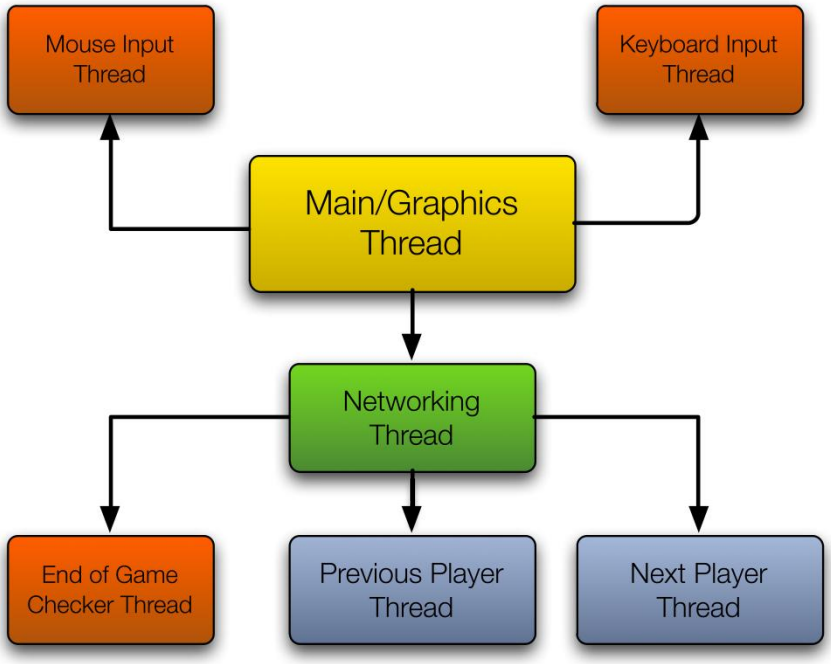
هر بازیکن پس از اجرا باید ریسه‌های زیر را ایجاد کند و منتظر برقراری ارتباط با بازیکن‌های قبلی و بعدی بماند:



ریسه‌ی مرتبط با بازیکن بعدی، در صورتی که موفق نشود که به بازیکن بعدی وصل شود، باید هر ۲ ثانیه یکبار به تلاش خود تا برقراری ارتباط ادامه دهد. استفاده از ریسه‌ی مرتبط با بررسی پایان بازی الزامی نیست، ولی استفاده از آن پایان صحیح بازی را تسریع خواهد بخشید.

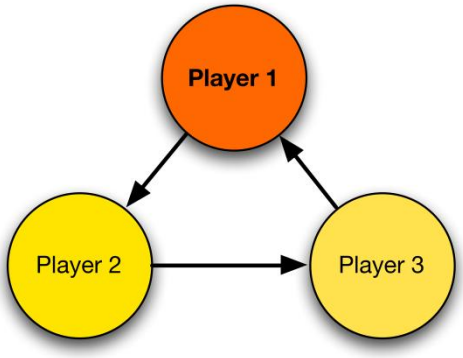
توجه داشته باشید که باید محیط گرافیکی OpenGL تنها در ریسه‌ی اصلی اجرا شود.

با در نظر گرفتن ریسه‌های مرتبط با ورودی (که توسط چارچوب‌کاری GLUT ساخته می‌شوند) و همچنین استفاده از یک ریسه‌ی کمکی برای جدا کردن بخش کارهای شبکه از بخش گرافیکی برنامه، در نهایت ساختار ریسه‌ای برنامه شما مشابه با شکل زیر خواهد شد:



مفهوم نوبت و زمان در بازی

در این بازی واحدهای زمان و نوبت مستقل از زمان واقعی سامانه است. برای مثال در شکل زیر سه بازیکن در بازی حاضر هستند. در صورتی که نوبت بازیکن ۱ باشد، تا بازیکن ۱ نوبت خود را بازی نکند، بازیکن‌های دیگر امکان بازی کردن ندارند. پس از بازی کردن، بازیکن ۱ با ارسال بسته‌ای حاوی تغییرهای نقشه به بازیکن ۲، نوبت خود را منتقل می‌کند و همچنین زمان را یک واحد افزایش می‌دهد.



نحوه‌ی بازی تیم‌های Soldier

تیم‌هایی که از نوع soldier باشند، دو حرکت می‌توانند انجام دهند:

Key Pressed	Action
u	Upgrade soldier
Enter	Create new soldier

در صورت انتخاب ارتقا سرباز، تمام سربازهایی که پس از این زمان توسط تیم حاضر ساخته شوند قابلیت‌های بهبود یافته‌ای خواهند داشت. در صورت ساختن سرباز جدید، یک سرباز در نقطه‌ی شروع مسیر بازی اضافه می‌شود.

حرکت سربازها به طور خودکار در زمان‌های ممکن باید صورت بگیرد. نحوه‌ی حرکت دادن سربازها در بخش «بروزرسانی بازی» آمده است.

نحوه‌ی بازی تیم‌های Tower

تیم‌هایی که از نوع tower باشند، دو حرکت زیر را می‌توانند انجام دهند:

Key Pressed	Action
u	Upgrade selected tower
Enter	Create tower in the selected place

برای هر کدام از عمل‌های گفته شده، بازیکن باید ابتدا یک خانه را با کمک واسط گرافیکی انتخاب کرده باشد، سپس با زدن یکی از کلیدهای گفته شده، انتخاب خود را نهایی کند.

ویژگی‌های بازیکن

هر کدام از بازیکن‌ها مشخصات زیر را دارند:

Property	Description
name	Team name
money	Team money (start value=20)
type	Soldier or Tower

بسته به نوع تیم، هر بازیکن مالک تعدادی سرباز یا برج است. ویژگی‌های هر یک در ادامه آمده است.

ویژگی‌های هر سرباز

سربازها ویژگی‌های زیر را دارند:

Property	Description
Owner	Team that created this soldier
Position	Current soldier's position
Life Capacity	Maximum Health (default=20)
Current Life	Current soldier's life
Pace	Number of cells traveled in each move (default=1)
Move Rate	Amount of time to pass to do a new move (default=2)

در صورتی که تیم سرباز، از نوبت خود برای ارتقا استفاده کند، ویژگی‌های زیر به صورت گفته شده برای سربازهای جدیدی که ساخته خواهند شد تغییر خواهد کرد:

Property	Upgrade Action
Pace	Increment
Move Rate	Decrement (if > 1)

ویژگی‌های هر برج

برج‌ها ویژگی‌های زیر را دارند:

Property	Description
Owner	Team that created this soldier
Position	Current soldier's position
Attack Rate	Amount of time to pass to do a new attack (default=3)
Damage	Amount of life consumed from the attacked soldier (default=10)

در صورت ارتقا یک برج ویژگی‌های زیر برای آن به شکل گفته شده تغییر می‌کند:

Property	Upgrade Action
Attack Rate	Decrement (if > 1)
Damage	Increment

هزینه‌ی عملیات

تمامی عمل‌های گفته شده موجب کسر پول از موجودی بازیکن می‌شود. در صورتی که موجودی بازیکن کافی نباشد، هیچ حرکتی رخ نمی‌دهد و نوبت فقط به بازیکن بعدی منتقل می‌شود. لیست هزینه‌ها در جدول زیر آمده است:

Action	Price
Create Tower	10
Upgrade Tower	5
Create Soldier	5
Upgrade Soldier	2

توجه داشته باشید که هر تیم فقط مجاز به انجام عمل‌های مرتبط با نوع خود است. برای مثال یک تیم از نوع soldier نمی‌تواند یک برج بسازد.

امتیازدهی و پایان بازی

یک تیم از نوع soldier با رسیدن هر سرباز به نقطه‌ی پایان، موجودی‌اش به میزان ظرفیت جان آن بازیکن افزایش می‌یابد. در مقابل یک تیم از نوع tower با کشتن هر سرباز، به میزان ظرفیت جان آن سرباز موجودی کسب می‌کند.

در صورتی بازی پایان می‌یابد که یک تیم بیش از حد بیشینه، موجودی کسب کرده باشد. در این صورت آن تیم برنده شده و تیم‌های دیگر بازنده می‌شوند. در صورتی که زمان بیش از حد مجاز گذشته باشد، بازی بدون داشتن برنده، پایان می‌یابد.

End of Game Criteria	Condition	Event
Money	**>= 50**	Win
Time	**>= 200**	Tie

بروزرسانی بازی

نوبت هر بازیکن که باشد، پس از انجام عمل مورد نظر، باید نقشه را با انجام مراحل زیر، به ترتیب ذکر شده، به‌روزرسانی کند و اطلاعات جدید بازی را به بازیکن بعدی منتقل کند:

۱. تمامی برج‌ها در صورتی که در زمان حاضر امکان حمله کردن دارند (با توجه به attack rate)، به یکی از سربازهای موجود در ۸ خانه‌ی اطراف خود حمله کنند. ترتیب حمله کردن برج‌ها و همچنین حمله به سربازها مهم نیست.

۲. تمامی سربازها در صورتی که امکان حرکت کردن دارند (با توجه به move rate)، به تعداد قدم‌های ممکن خود (میزان pace) در مسیر تعیین شده در نقشه جلو برده شوند.

۳. در مراحل فوق، در صورت کشته شدن یک سرباز و یا رسیدن یک سرباز به خط پایان، امتیاز تیم مرتبط افزایش یابد. در صورت کشته شدن و یا به خط پایان رسیدن یک سرباز، از نقشه حذف شود.

۴. در صورتی که تیمی برنده شد، بازی پایان یابد.

۵. زمان یک واحد افزایش پیدا کند.

۶. نوبت به بازیکن بعدی منتقل شود. تمام اطلاعات محل و ویژگی‌های تمام برج‌ها و سربازها به همراه زمان کنونی باید به بازیکن بعدی انتقال داده شود.

نمونه‌ی اجرای بازی

در ادامه نمونه‌ای از اجرای بازی آمده است: (فقط برخی از ویژگی‌ها بررسی شده‌اند)

Game Element	Move Rate
Soldier1	**3**

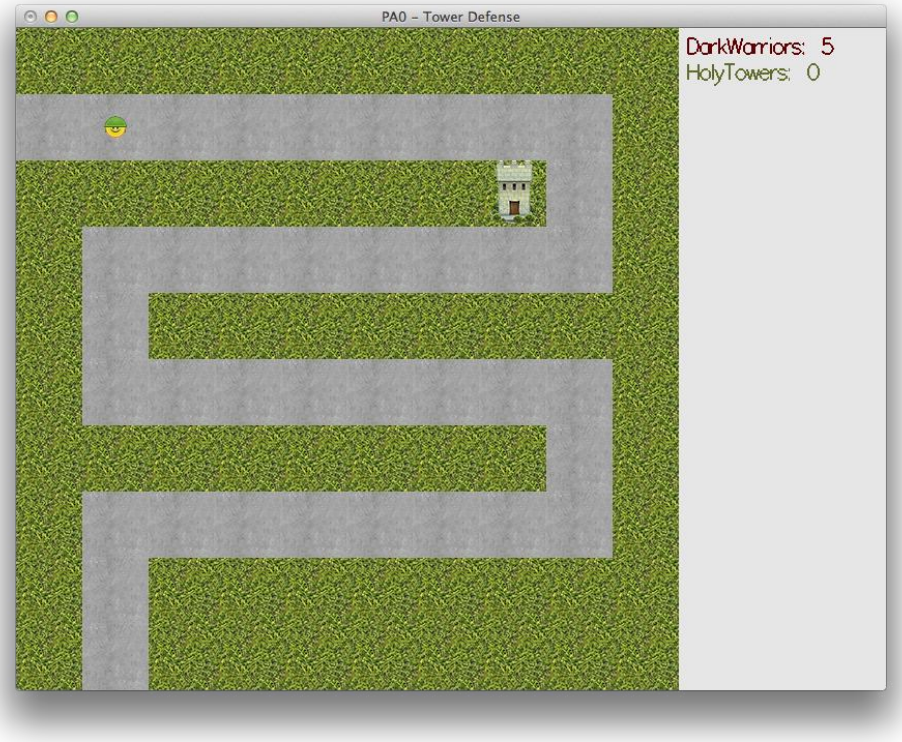
Game Element	Attack Rate
Tower1	**5**

اجرای بازی:

Time	17	18	19	20	21	22	23	24	25
Soldier1		Move			Move			Move	
Tower1				Attack					Attack

گرافیک بازی

برای نمایش بازی باید از کتابخانه‌ی OpenGL استفاده کنید. نمونه‌ی کد و کدهای اولیه‌ی مورد نیاز و همچنین فایل‌های عکس بازی در اختیارتان قرار خواهد گرفت. هدف نهایی نمایش وضعیت بازی و همچنین امکان انتخاب یک خانه‌ی خاص و انجام عملیات هر بازیکن از طریق این واسط گرافیکی است. نمونه‌ای از چنین واسطی را در زیر می‌توانید ببینید:



نمونه‌ای از یک رابط گرافیکی کامل

واسط گرافیکی برای تیم‌های از نوع tower باید امکان انتخاب یک خانه را داشته باشد. برای اینکار از توابع پایه‌ی OpenGL و همچنین GLUT که در بخش ضمیمه توضیح داده شده‌اند، می‌توانید استفاده کنید. نمونه‌ای از نحوه‌ی نمایش خانه‌ی انتخاب شده در صفحه‌ی بعد آمده است.

در نهایت این واسط باید نقشه‌ی به‌روز شده، تیم‌ها و موجودی آن‌ها و اینکه نوبت این بازیکن هست یا خیر را نمایش دهد.
همچنین امکان تعامل برای انتخاب عمل مورد نظر را بدهد.

نمونه‌ای از نحوه‌ی نمایش خانه‌ی انتخاب شده

نکات ضروری

- برای آشنایی بیشتر با کتابخانه‌ی OpenGL می‌توانید کتاب زیر را مطالعه کنید:
  http://www.glprogramming.com/red/
- برای آشنایی با نحوه‌ی ساخت thread با کتابخانه‌ی POSIX thread به آدرس زیر مراجعه کنید:
  http://www.yolinux.com/TUTORIALS/LinuxTutorialPosixThreads.html
- در صورتیکه هر مشکل یا پرسشی داشتید که فکر می‌کنید پاسخ آن برای همه مفید خواهد بود، لطفاً آن را به گروه پستی درس ارسال کنید.
- به هیچ عنوان جواب تمرین را به گروه پستی نفرستید.
- فرستادن کل یا قسمتی از برنامه‌تان برای افراد دیگر، یا استفاده از کل یا قسمتی از برنامه‌ی فردی دیگر به نام خود، تقلب محسوب می‌شود.
- پس از اتمام کارتان لازم است که فایل‌های خود را به همراه Makefile فشرده کرده و بر روی سامانه‌ی خودکار داوری[۲] upload کنید.

موفق باشید

[2] http://partov.sharif.edu/

ضمیمه ۱: آشنایی با کتابخانه‌ی OpenGL و GLUT

کتابخانه‌ی OpenGL مشهورترین کتابخانه‌ی گرافیکی است که در همه‌ی platformها امکان استفاده از آن وجود دارد. در این ضمیمه توابعی که نیاز شما را برای نمایش واسط گرافیکی رفع می‌کنند، توضیح داده شده‌اند. نمونه‌ی استفاده از این توابع در نمونه کدی که در اختیار شما قرار می‌گیرد، موجود است.

نام تابع	کارکرد
glColor3f	تغییر رنگ برای عملیات کشیدن بعدی. در صورتی که جسمی کشیده شود که دارای بافت باشد، رنگ تعیین شده توسط این تابع با بافت جسم ترکیب می‌شود. با تعیین رنگ سفید، چنین ترکیبی رخ نخواهد داد.
glBindTexture	برای انتخاب یک بافت برای هر سطحی که در ادامه کشیده شود، استفاده می‌شود. یک بافت به طور معمول یک عکس بارگذاری شده در حافظه‌ی OpenGL است.
glTranslatef	برای جابجایی مختصات دنیا استفاده می‌شود. دقت کنید که مختصات آن نسبی است و پس از فراخوانی اول، فراخوانی‌های بعد نسبت به مکان جدید خواهند بود.
glutPostRedisplay	پس از تغییر در اجزای صفحه، با فراخوانی این تابع می‌توانید صفحه را از نو بکشید.

ضمیمه ۲: نکاتی برای استفاده از کد پایه

برای شروع کار با کتابخانه‌ی OpenGL، نمونه کدی در اختیار شما قرار خواهد گرفت. یک برنامه‌ی نمونه در فایل simple.cpp نوشته شده است. این برنامه با استفاده از کتابخانه‌ی OpenGL یک صفحه‌ی ابتدایی را می‌کشد. علاوه بر کدهای داده شده، چهار فایل عکس با فرمت Targa برای استفاده در واسط گرافیکی فراهم شده است.

برای کامپایل کردن و اجرای برنامه باید ابتدا کتابخانه‌ی OpenGL و GLUT را نصب کنید.

سامانه‌ی عامل Linux

اگر از Ubuntu استفاده می‌کنید، دستور زیر را در کنسول وارد کنید:

```
sudo apt-get install libglu1-mesa-dev freeglut3-dev mesa-common-dev
```

سپس برای کامپایل برنامه با کمک کتابخانه‌ی GLUT دستور زیر را بزنید:

```
g++ simple.cpp -lglut -o simple
```

اگر از distribution دیگری استفاده می‌کنید، به جای apt-get از دستور معادل آن برای نصب packageها استفاده کنید.

سامانه‌ی عامل Mac OS X

در صورتی که از سامانه‌ی عامل Mac OS X استفاده می‌کنید، با نصب Xcode command-line tools کتابخانه‌ها و دستورهای لازم را برای کامپایل برنامه خواهید داشت. می‌توانید با دستور زیر کد را کامپایل کنید:

```
g++ -framework OpenGL –framework GLUT simple.cpp -o simple
```

توجه: در تحویل حضوری، کد شما فقط با سامانه‌ی عامل لینوکس تست و اجرا می‌شود. در صورتی که از سامانه‌ی عامل لینوکس برای زدن کد استفاده نمی‌کنید، قبل از ارسال تمرین از کار کردن کدتان در این سامانه‌ی عامل اطمینان حاصل کنید.

اجرای کد پایه

پس از کامپایل کردن، با اجرای برنامه به خروجی‌ای مشابه عکس زیر باید برسید:

خروجی حاصل از اجرای کد پایه

نکاتی پیرامون سرآیندهای کد پایه

۱. کار اصلی کشیدن واسط گرافیکی در تابع draw در فایل simple.cpp صورت می‌پذیرد. برای بروزرسانی صفحه باید با تابع glutPostRedisplay دوباره صفحه را بکشید. توجه داشته باشید که امکان بروزرسانی فقط یک بخش از صفحه وجود ندارد و باید تمام صفحه را دوباره با تابع draw بکشید.

کتابخانه‌ی OpenGL، مانند اکثریت کتابخانه‌های گرافیکی، فقط در یک ریسه باید کار کند. به همین دلیل از صدا زدن تابع‌های آن از ریسه‌های دیگر بپرهیزید. تنها یک استثنا وجود دارد و آن تابع glutPostRedisplay است که جزو چارچوب کاری GLUT است. این تابع را از هر ریسه‌ی دیگری می‌توانید صدا بزنید و دستور به روزرسانی صفحه در یک صف قرار خواهد گرفت و در نهایت دستور کشیدن مجدد صفحه به همان ریسه‌ی اصلی OpenGL فرستاده خواهد شد.

دو تابع mouse و keyboard برای کنترل ورودی در فایل simple.cpp نوشته شده‌اند. هر کدام از این توابع هنگام وارد شدن یک ورودی جدید توسط کاربر، در یک ریسه‌ی جداگانه برای پردازش ورودی، صدا زده خواهند شد. این دو تابع در فایل graphics.h توسط دو تابع glutMouseFunc و glutKeyboardFunc تعیین شده‌اند.

۲. فایل graphics.h حاوی تابع‌های لازم برای آماده‌سازی محیط OpenGL است. در صورت نیاز برای تغییر عکس‌های مورد استفاده می‌توانید فایل‌های استفاده شده را در تابع init تغییر دهید. فقط از عکس‌هایی با فرمت Targa می‌توانید استفاده کنید.

۳. فایل map.h حاوی تابع‌های و تعریف‌های اولیه‌ای برای راحتی کشیدن نقشه در OpenGL است. به طور مشخص تابع draw_cell یک مربع را با اندازه‌ی ضلع مشخص در صفحه می‌کشد. نقطه‌ی شروع کشیدن را با تابع glTranslatef می‌توانید تغییر دهید. تابع glprint هم در این فایل برای کشیدن یک نوشته در صفحه استفاده می‌شود. نمونه‌ی استفاده شده‌ی این توابع را در فایل simple.cpp می‌توانید مشاهده کنید.

۴. در نهایت فایل log.h برای گرفتن log از برنامه است. کلاس تعریف شده در این فایل thread-safe است. استفاده همزمان از cout از ریسه‌های مختلف ممکن است شی مرتبط با آن را خراب کند. علت این امر نبودن موجودیت ریسه در استاندارد اولیه‌ی زبان C++ است. در نتیجه جلوگیری از استفاده همزمان از این شی خاص که در همه‌ی ریسه‌ها به اشتراک گذاشته شده است، ضروری است.

شیوه‌ی استفاده از آن مانند cout است:

```
LOG << "a = " << a << endl;
```

دو شی DEBUG و ERR هم در این فایل وجود دارند. این دو شی در صورت گرفتن خروجی در کنسول، خروجی را به رنگ زرد و قرمز نمایش می‌دهند.

در صورتی که می‌خواهید خروجی در فایل ذخیره شود، خط حاوی #define FILE_LOGGING را از کامنت خارج کنید. با اینکار برای هر اجرای برنامه یک فایل با نام تصادفی برای ذخیره‌ی log ساخته می‌شود (تصادفی بودن نام، برای جلوگیری از تداخل نام فایل، هنگام اجرای همروند چند نمونه از یک فایل باینری است).