پروژه برنامه نویسی پیشرفته فاز چهارم

ابتدا نقاط ضعف و قوت پروژه را ذکر میکنیم. سپس منابعی که از آن استفاده شده را ذکر می‌کنیم.

نقاط قوت :

۱- اولین نکته مهم این است که برنامه کاملا شی گرا است و سعی شده پکیج بندی ها و کلاس بندی ها با نهایت دقت انجام شود و سطح دسترسی تابع ها و متد ها حداقل قرار داده شده. پکیج اصلی بازی hearthstone است که خود دارای چند پکیج است.

۲- دومین مورد گرافیک برنامه است که سعی شده زیبا طراحی شده و همه دکمه ها و آیکون ها ساخته شده باشند و مستقیم از خود کتابخانه سوییِنگ گرفته نشوند. (که البته در این بخش از یکی از دوستان کمک گرفتیم که در بخش منابع ذکر شده است)

۳- سومین مورد استفاده از جلوه‌های صوتی زیبا برای صفحه اصلی و صفحه بازی (به صورت جداگانه) است. البته برای کارت ها هنگام بازی شدن روی زمین هم صداهایی قرار داده شده. همچنین برای دکمه end turn، و صحنه آخر بازی که برد و باخت را مشخص می‌کند هم صداهای مناسب قرار داده شده.

۴- همچنین تمام اجزای بازی اعم از هیرو ها و کارت ها به آسانی قابل تغییر هستند. یعنی ما قالب کارت یا هیرو یا حتی یک دکمه را می‌گیریم، سپس متن یا تعداد مانا یا خون یا ... را روی آن می‌نویسیم. البته این کار در فاز قبل انجام شد و در این فاز توسعه یافت. برای مثال برای هیرو پاور ها یا اسلحه ها(و یا حتی اسلحه های ارتقا یافته!)

۵- در هر مرحله ارور های درستی به کاربر نشان داده شده است. در حین بازی هم تمامی ارور ها به کاربر نشان داده می‌شوند. برای مثال اگر با یک کارت rush به محض کاشته شدن قصد حمله به هیرو را داشت، هشدار لازم داده می‌شود.

۶- اطلاعات کانفیگ برای گرافیک و اطلاعات بازی در دو فایل جدا و در بیرون از بازی ذخیره شده‌اند و در بیرون از برنامه نگه‌داری می‌شوند.

۷- لاجیک بازی با استفاده از اینترفیس های گوناگون زده شده و شبیه پترن Observer می‌باشد، اما اندکی تفاوت وجود دارد! یعنی ما صرفاً برای موجودات اینترفیس ها یا رفتار آن‌ها را تعریف میکنیم، و در صفحه بازی موقع لزوم این رفتار ها بروز پیدا می‌کنند!

۸- انواع مختلفی برای موس طراحی شده. برای مثال حالت موس برای هیل، حمله، فریز کردن و دنبال چیزی گشتن فرق می‌کند. و همه موارد ذکر شده با حالت معمولی موس متفاوت هستند. این کار به کاربر کمک می‌کند متوجه شود اکنون در حال انجام چه کاری است. همچنین با راست کلیک می‌توان عملیات فعلی را کنسل کرد. یعنی اگر در حال حمله هستیم، با زدن راست کلیک به حالت عادی بر می‌گردیم.

۹- همچنین کلاس‌هایی برای ارتباط با بیرون از برنامه درست شده اند. کار این کلاس این است که اطلاعات را از resource برنامه بخوانند. البته اگر قبلاً این اطلاعات خوانده شده بود، دوباره خوانده نمی‌شود و همان اطلاعات قبلی را بر می‌گردانند. این کار باعث شد مشکل سرعتی که در فاز قبل داشتیم حل شود و سرعت برنامه چندین برابر شد.

۱۰- در این فاز از انیمیشن های زیبا استفاده شده که این کلاس‌ها داینامیک بودند و قابلیت استفاده مجدد را دارند. البته انیمیشن ها یک مرحله‌ای هستند. ولی به گونه‌ای طراحی شدند که می‌توان مسیر یک انیمیشن را در حین حرکتش تغییر داد!

۱۱- همچنین بسیاری از ساختار های فاز های قبل که از اصول SOLID پیروی نمی‌کردند، در این فاز تغییر کردند. البته بعضی هم به علت کمبود وقت، تغییر خاصی نداشتند!

۱۲- در این فاز از سیستم درخواست و پاسخ برای ارتباط بین کلاینت و سرور استفاده شده. یعنی یک مپر در سرور و یک مپر در کلاینت وجود دارد. که تنها راه ارتباط بین کلاینت و سرور، ارتباط بین این دو مپر است. هر کدام از مپر ها، دارای تعداد تابع با نام درخواست یا پاسخ (البته موارد خاصی هم مشاهده می‌شود. اما فرم کلی به همان صورتی است که گفته شد)، به این صورت که یک کلاس به نام پکت ساخته شده و تنها همین کلاس بین کلاینت و سرور فرستاده می‌شود. این کلاس، دارای ۲ متغیر است. یک رشته به عنوان اسم یک تابع و یک لیست از آبجکت ها برای ورودی های تابع است.

به این‌صورت که تابعی را که می‌خواهیم در سمت دیگر شبکه اجرا کنیم، نام تابع، به همراه ورودی های تابع را در یک کلاس پکت قرار می‌دهیم و این کلاس پکت را به سمت دیگر می‌فرستیم. (اگر در سرور هستیم، می‌خواهیم تابعی را در کلاینت اجرا کنیم. یا برعکس)

در آخر از نام تابعی که در کلاس پکت هست، به کمک رفلکشن، تابع مورد نظر را با مقادیر ورودی مورد نظر اجرا می‌کنیم.

۱۳- قابلیت مشاهده بازی توسط دیگران و همچنین دیدن بینندگان بازی توسط دو بازیکن زمین در این فاز اضافه شده. به علاوه قابلیت حذف بینندگان توسط بازیکن های زمین نیز افزوده شده.

۱۴- در صفحه هایی که اطلاعات اولیه آن‌ها باید از سرور گرفته شود، یک صفحه loading نمایش داده می‌شود. مادامی که اطلاعات از سرور با موفقیت گرفته شد، شما وارد صفحه مورد نظر خواهید شد!

۱۵- برای اتصال بازیکن ها در بازی آنلاین، شرایطی در نظر گرفته شده که در ادامه توضیح خواهیم داد.

۱۶- برای امنیت بازی هم، در حین بازی، اطلاعات اضافی به کاربر ها نمی‌دهیم. یعنی کاربری نمی‌تواند با تغییر دادن کد، به کارت های دست حریف دست پیدا کند. یا اینکه رفتار کارت های خودش را در حین بازی تغییر دهد. چون لاجیک بازی در سرور است و کلاینت صرفاً مسئول نمایش اطلاعات است.

۱۷- در آخر نقاط قوتی که از فاز های قبل داشتیم هم حفظ شده اند. برای مثال هش شدن پسوورد ها و ذخیره نکردن خود پسوورد اکانت ها هم در این فاز رعایت شده.

نقاط ضعف :

۱- قسمت اطلاعات کامپوننت های گرافیکی بسیار زیاد است، این به خاطر این است که هر کامپوننت را در جای مشخصی قرار داده‌ایم و چون امکانات بهتری در اختیار نبود، برای همه کامپوننت ها مختصاتشان را در اول هر کلاس به صورت دو متغیر استاتیک فاینال نوشته ایم. البته دقت کنید این مقادیر کانفیگ نیستند. چون دارای محاسبات ریاضی هستند و مکان هر کامپوننت بر حسب مکان بقیه کامپوننت ها تعیین می‌شود.

۲- سیستمی که برای گرافیک بازی زده شده، به اندازه کافی داینامیک نیست و قابلیت‌هایی مثل ریسایز کردن را ندارد!

۳- بعضی از اشکالاتی که در فاز های قبل انجام شده بود، به دلیل کمبود وقت در این فاز تغییر نکردند. برای مثال کار کردن با آبجکت هنوز زیاد است. برای مثال، خیلی از جاها می‌توانستیم به جای اینکه خود کارت را نگه داریم، صرفاً اسم کارت را نگه داریم. ولی این‌کار به تغییرات بنیادی نیاز داشت که متأسفانه به علت کمبود زمان به همان شکل باقی ماندند.

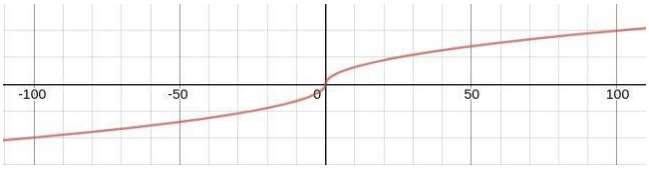
۴- به علت نحوه کدزنی که در فاز های قبل داشتیم، مجبور به ذخیره کردن بعضی از اطلاعات غیرضروری در دیتابیس شدیم. مثلاً برای هر اکانت جدید که ساخته می‌شود، در دیتابیس، ۵ قهرمان جدید اضافه می‌شوند.

۵- در مورد زمانی که هر بازیکن برای بازی کردن دارد (در این بازی، ۶۰ ثانیه در نظر گرفته شده)، این تایمر در کلاینت پیاده شده. بنابراین یک کاربر می‌تواند با تغییر دادن قسمت کلاینت، تا بی نهایت فرصت بازی داشته باشد که به علت کمبود وقت متأسفانه نتوانستیم این مورد را درست کنیم.

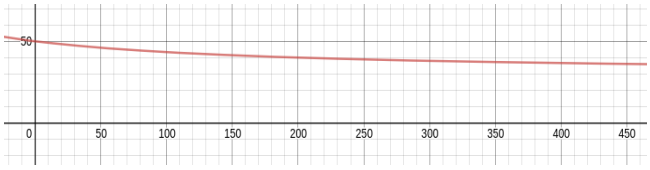
توضیحات :

برای محاسبه تغییر جام یک بازیکن در یک بازی، به موارد زیر توجه می‌کنیم :

- اگر جام کنونی بازیکن اول برابر با a و تعداد جام کنونی بازیکن دوم برابر با b باشد، آنگاه تعداد جام های بازیکن برنده به اندازه g(a + b) + f(a − b) زیاد می‌شود و تعداد جام های بازیکن بازنده به همین اندازه کم می‌شود.
- تابع f وظیفه این را دارد که اگر یک بازیکن از یک بازیکن خیلی ضعیف‌تر (یا خیلی قوی‌تر) از خودش برد (یا باخت)، جام به اندازه کم‌تری (یا بیشتری) از حالت عادی تغییر کند.
- تابع g هم وظیفه اش این است که در مقدار جام های کم، تعداد جام ها سریع تر زیاد شود و در مقدار جام زیاد، آهسته تر زیاد شود.
- پس تابع g همواره مثبت است ولی تابع f ممکن است منفی هم باشد. شکل این توابع را در زیر می‌بینید:



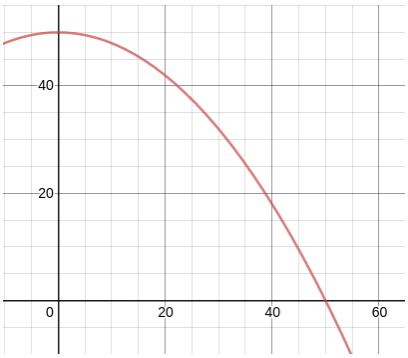
تابع f



تابع g

در مورد الگوریتم وصل کردن دو بازیکن به یکدیگر، موارد زیر در نظر گرفته شده :

- اختلاف جام های دو بازیکن نباید بیشتر از ۱۰۰ باشد. در غیر این‌صورت، دو بازیکن هیچوقت نمی‌توانند با یکدیگر بازی کنند.
- به ۵۰ بازی اخیر دو بازیکن که شرط بالا را دارند نگاه می‌کنیم. در هر بازی مقداری جام اضافه یا کسر شده است. یک تابع سهمی طور روی این ۵۰ بازی اعمال می‌کنیم و یک عدد به دست می‌آوریم. سهمی بودن تابع به این خاطر است که هر چه به بازی های قدیمی‌تر نگاه می‌کنیم، ارزش آن بازی کم تر است.
- حال اختلاف این دو عددی که به بازی‌های اخیر هر بازیکن نسبت داده ایم، نباید خیلی زیاد باشد. البته زمان هم در این مرحله تاثیر گذار است. یعنی اگر اختلاف زیاد بود و زمان زیادی از درخواست بازیکن اول گذشته بود، این بازی برقرار می‌شود. چون بالاخره دو بازیکن شرط اول یعنی نزدیک بودن جام به هم را دارا هستند.
- شکل تابع سهمی را هم در زیر می‌بینید. این که سهمی رو به پایین است، به این دلیل است که بازی های اخیر از راس سهمی شروع می‌شوند. به این صورت که اخرین بازی، در راس سهمی است. بازی یکی مانده به آخر، در نقطه ای به طول یک از سهمی است و ...



تفاوت هایی که با بازی اصلی و شاید داک توضیحات شاهد هستیم :

- اول اینکه تعریف کارت قفل کارتی است که کاربر نمیتواند آن را استفاده کند (ممکن است آن را داشته باشد ولی قابل استفاده نیست). پس شما نمی‌توانید یک کارت قفل را به دک خود اضافه کنید!
- مورد دوم اینکه هر قهرمان تعدادی دک برای خود دارد.

برای شروع بازی ابتدا باید اکانت خود را بسازید. سپس به بخش کالکشن رفته و قهرمان خود را انتخاب کنید. برای قهرمان خود یک دک بسازید و دک خود را بچینید. (البته باید دک خود را انتخاب هم بکنید!) حال شما آماده بازی هستید!

یک حالت دیگر از بازی، deck reader می‌باشد. در این حالت شما نیازی به چیدن دک ندارید. صرفاً این گزینه را انتخاب کنید، حال سرور شما را به یک نفر دیگر که همین حالت را انتخاب کرده وصل می‌کند و توسط یک دک که در سرور قرار دارد، شما می‌توانید با یکدیگر رقابت کنید!

Java Version : 11

Project Build Tool : Maven 3.6.3

Database: PostgreSQL

Dependencies :

com.fasterxml.jackson.core : jackson-core : 2.11.1
com.fasterxml.jackson.core : jackson-annotations : 2.11.1
com.fasterxml.jackson.core : jackson-databind : 2.11.1
org.postgresql : postgresql : 42.2.14
org.hibernate : hibernate-core : 5.4.14.Final

- دلیل استفاده از جکسون در این فاز این بود که آبجکت مپر جکسون، می‌تواند نوع داده‌ها را ذخیره کرده و موقع بازگردانی از آن استفاده کند.

منابع استفاده شده :

GeeksforGeeks

Stack Overflow

LunaPic

Online Audio Converter

Image Online – Crop a Circle Tool

قسمتی از کد برای خواندن کلاس از یک فایل جَر، از این آدرس کپی شده است.

در آخر تشکر میکنیم از سید مجتبی مدرسی (@Xamarin_Developer) بـرای کمـک هـایش در دیـزاین و گرافیـک برنامه. به طوری که گرافیک زیبای برنامه رو مدیون ایشون و مشاوره هاشون هستیم.

Designed by AliTavassoly & XamarinDev