گزارش پروژه‌ی بازی

در این پروژه، 4 فایل کد و 1 فایل برای ذخیره‌ی داده‌های بازی و 6 عکس داریم.

فایل های کد عبارتند از: game_client , game_server , game_window , game_characters

فایل ذخیره: information

توضیحات:

(game_server)

ابتدا در سرور فایل information را همراه با زمان اجرای سرور ایجاد میکنیم. با استفاده از کتابخانه‌ی game_characters که در سرور import شده، کلاس شخصیت های بازی را میسازیم. یک سوکت میسازیم و با استفاده از حلقه‌ی for منتظر می شویم دو client به آن وصل شوند.

تابع make_info_message_about_character با ورودی های کلاس کاراکتر، میزان جان هر کاراکتر و میزان جادو، اطلاعات هر کاراکتر را می سازد . بعد از اتصال، تابع sending_character_info اطلاعات را از تابع بالا گرفته و رشته ها را با یک ستاره به یکدیگر متصل کرده و برای ورودی خود (client) میفرستد.

مقدار اولیه 100 را برای جان و میزان جادو هر دو بازیکن و راند ابتدایی را 1 در نظر میگیریم.

در while، از try except استفاده میکنیم. در قسمت try یک دیکشنری خالی با نام dic_client_message ایجاد میکنیم. سپس با استفاده از for تابع recving_client_choice را اجرا میکنیم و مقدار های دریافتی از بازیکن ها در dic_client_message ذخیره میشود و در صورتیکه خروجی این تابع "the end" بود، raise Exception میکنیم و سوکت سرور را بسته و با دستور break از while خارج میشویم.

تابع recving_client_choice : مقدار عددی به عنوان ورودی میگیرد که برابر ایندکس لیست clients است. سپس از عضو با ایندکس داده شده اطلاعات میگیرد و در صورتیکه این مقدار دریافتی برابر "end" بود رشته‌ی "the end" را برمیگرداند و در غیر این صورت، به دیکشنری با کلید عدد ورودی مقدار دریافتی را اضافه میکند.

با استفاده از دیکشنری ساخته شده دو لیست از انتخاب بازیکن‌های میسازیم که شامل نام کاراکتر و حرکت منتخب هر بازیکن است. تابع calculate_life حرکت دو بازیکن و اسم کلاس کاراکتر آنها (که از دیکشنری گرفته شده) را به عنوان ورودی دریافت کرده و خروجی مقداری است که از جان بازیکنی که اطلاعاتش اول وارد شده، کم می شود. تابع calculate_magic نیز به همان شکل کار میکند اما مقدار خروجی آن مربوط به جادو است. از این دو تابع استفاده میکنیم تا تغییرات جان و جادوی بازیکن ها را محاسبه کنیم.

اگر مقدار جان هردو، همزمان تمام شود، برای هردو پیام تساوی ارسال میشود. اگر مقدار جان یکی از آن دو تمام شود برایش پیغام باخت و برای دیگری پیغام برد ارسال میشود. در صورتیکه جان هیچکدام تمام نشد، با استفاده از تابع make_info_meassage_about_character یک رشته‌ی اطلاعاتی برای بازیکن ها ارسال میشود. در ادامه تابع write_information_in_file تغییرات جان و جادوی بازیکن ها را در یک فایل ذخیره میکند. در انتها، در صورتیکه جان یکی از دو بازیکن تمام شده باشد، مقدار های اولیه را دوباره به برنامه میدهیم و در صورت بازی مجدد بازیکن ها، مقدار جان و جادو از 100 کم میشود. (این چرخه تا زمانی که رشته‌ی "the end" دریافت شود تکرار میشود)

(game_client)

یک سوکت می سازیم و آن را با دستور connect به سرور متصل میکنیم. منتظر میشویم تا رشته‌ی اطلاعات کاراکتر هارا به ما بدهد. این رشته را با "*" جدا میکند و با یک دستور for اطلاعات هر کاراکتر را با کلید اعداد 1 تا 6 داخل دیکشنری characters_info_dic میریزد. سپس تابع start_window را از کتابخانه‌ی game_window اجرا کرده و به آن، دیکشنری و سوکت ساخته شده را میدهد. بعد از نمایش صفحات گرافیکی به بازیکن ها و انتخاب هایشان، در تابع recving_results_from_server که در یک while اجرا شده و در صورت بروز خطا با دستور break از آن خارج میشود، اطلاعات را از سرور دریافت میکند. در تابع بالا در صورت اعلام اتمام بازی از سرور، تابع showing_results اجرا میشود. در غیر این صورت تابع choose_your_move برای انتخاب مجدد حرکت توسط بازیکن اجرا میشود.

(game_characters)

یک کلاس magician تعریف کردیم که با گرفتن 5 ورودی، در کل 11 ویژگی دارد.

تابع random_selection: این تابع یک مقدار عدد ورودی گرفته و با استفاده از کتابخانه‌ی رندوم و دستور choices بین 0 و 1 با وزن های (عدد-100) و (عدد)، صفر یا یک را برمیگرداند.

تابع calculate_life : در این تابع ما کاهش جان اولین بازیکن داده شده را بررسی میکنیم. مقدار پیشفرض damage=0 دارد. اگر بازیکن اول گزینه‌ی دفاع را انتخاب کرده باشد و احتمال وقوع دفاع (که با تابع random_selection بدست آمده) صفر باشد و بازیکن دوم حمله موثر(از نوع جادو یا ضربه) داشته باشد بسته به نوع حمله مقداری پیشفرض با توجه به کلاس کاراکتر برگردانده میشود.(این مقدار از جان بازیکن اول کم می شود)

اگر بازیکن اول حمله (از نوع جادو یا ضربه) را انتخاب کرده باشد و بازیکن دوم گزینه حمله (از نوع جادو یا ضربه) را انتخاب کرده باشد و این حمله موثر باشد، بسته به نوع حمله مقداری پیشفرض با توجه به کلاس کاراکتر برگردانده میشود.(این مقدار از جان بازیکن اول کم می شود)

تابع calculate_magic: در این تابع ما کاهش جادو اولین بازیکن داده شده را بررسی میکنیم. مقدار پیشفرض decrease=0 دارد.

اگر حرکت بازیکن اول حمله جادویی بود و مقدار تغییر جان بازیکن اول مخالف صفر بود، decrease برابر است با شدت حمله منهای مقداری که به ازای موثر بودن حمله‌ی بازیکن اول قرار است به جادوی وی اضافه شود.

اگر حرکت بازیکن اول حمله‌ی جادویی بود و مقدار تغییر جان بازیکن اول مساوی صفر بود، decrease برابر است با شدت حمله.

(game_window)

تابع start_window : یک پنجره با نام battle of the villans و نوشته‌ی نام بازی و welcome را نشان میدهد و در انتها یک دکمه‌ی start داریم که تابع choose_window و close_window را به ترتیب اجرا میکند.

تابع close_window: نام پنجره را از ورودی میگیرد و آن را از بین میبرد.

تابع choose_window : دکمه هایی با نام های کاراکتر ها و دکمه خروج نمایش داده میشود که با کلیک به روی اسم کاراکتر ها تابع show_character_info اجرا میشود که ورودی آن به ترتیب دیکشنری، کلید یکی از اعضای دیکشنری، client هستند. با کلیک روی دکمه‌ی خروج تابع close_window و client_exit اجرا میشود.

تابع make_dic_of_names: ورودی دیکشنری میگیرد که برای هر value آن تابع take_out_the_name را اجرا میکند و مقدار خروجی آن را به list_name اضافه میکند. درنهایت این لیست را برمیگرداند.

تابع take_out_the_name : ورودی رشته میگیرد و آن را با \n جدا میکند. عضو اول لیست بدست آمده را با ":" جدا میکند و عضو دوم لیست ساخته شده را برمیگرداند.

تابع show_character_info: اطلاعات و عکس کاراکتر انتخاب شده و دکمه های بازگشت، انتخاب کاراکتر، خروج را نمایش میدهد. با انتخاب کاراکتر توابع close_window و choose_your_move اجرا میشود. با دکمه بازگشت توابع close_window و choose_window اجرا میشود. با دکمه‌ی خروج توابع close_window و client_exit اجرا میشود.

تابع client_exit: Goodbye را نمایش میدهد و رشته‌ی "end" را برای سرور میفرستد و client را میبندد.

تابع choose_your_move: پنجره باز میشود که اسم کاراکتر، 5 دکمه برای انتخاب حرکت، دکمه‌ی خروج ، 2 نوار برای نمایش میزان جان و جادو را نمایش میدهد. در صورت کلیک روی هر یک از دکمه های حرکت، توابع close_window و

client_selection اجرا میشود. ورودی تابع client_selection حرکت انتخابی، رشته اطلاعات کاراکتر و سوکت client است.

در صورت کلیک روی دکمه خروج، توابع close_window و client_exit اجرا میشود. سپس تابع take_out_life_and_magic اجرا میشود و جان و جادو را بازمیگرداند.

برای ساختن نوار های نشان گر جان و جادو، از تابع space به عنوان نوشته‌ی داخل لیبل استفاده کردیم.

تابع space: به عنوان ورودی عدد میگیرد و با یک for به تعداد آن عدد فاصله قرار میدهد و آن رشته را برمیگرداند.

تابع take_out_life_and_magic : ورودی رشته اطلاعات کاراکتر میگیرد. رشته را با \n جدا میکند. عضو دوم و سوم لیست بدست آمده را با ":" جدا میکند و عضو دوم لیست های نهایی آنها را بازمیگرداند.

تابع client_selections : ورودی حرکت و اطلاعات کاراکتر و سوکت client را میگیرد. با استفاده از تابع take_out_the_name و اطلاعات کاراکتر، اسم کاراکتر را به ما میدهد. اسم کاراکتر و حرکت را با "*" به هم وصل کرده و به سرور میفرستد.

تابع showing_results: ورودی دیکشنری، رشته ی نتیجه‌ی بازی و سوکت client است. رشته‌ی نتیجه را نمایش میدهد. دو دکمه خروج و بازی مجدد دارد. با انتخاب بازی مجدد تابع close_window و choose_window اجرا میشود. با انتخاب خروج تابع close_window و client_exit اجرا میشد.

پروژه‌ی گروه حسنا سلطان الکتابی (4014013055) و یگانه رستگاری کوپائی (4014013040)