Documentazione del progetto gioco di carte degli studenti Gabriel Rodriguez e Andrea Ravanetti.

CARTA.JS

contiene gli attributi che interessano la classe carta e le funzioni per creare delle carte preimpostate.

DECK.JS

contiene come attributo un array che poi sarà riempito di carte, ha al suo interno le funzioni: mescola per mescolare in maniera pseudo random il mazzo,

controllo per spostare le carte dal cimitero al mazzo e poi mescola il mazzo automaticamente,

svuota che resetta il deck a un array vuoto,

aggiungiCarta che aggiunge una carta passata alla funzione al mazzo,

creaDeckBuono che crea un mazzo predefinito per il giocatore,

creaDeckCattivo che crea un mazzo predefinito per il computer.

ENERGYMANAGER.JS

ha come variabili globali un valore associato a quanto sono piene le barre della stamina, e ha funzioni per inizializzare, aggiornare e resettare le barre della stamina.

HAND.JS

È una classe che contiene un array che poi sarà riempito dalle carte che dovranno essere in mano, ha le funzioni per pescare, rimuovere una carta e spostarla al cimitero e un'altra per rimuovere una carta specifica.

LIFEMANAGER.JS

Ha come variabili globali valori associati a quante sono piene le barre della vita e dell’armatura, ha funzioni per settare le barre a valori specifici, funzioni per aumentare o diminuire le barre in base all’intero passato. Queste funzioni hanno al loro interno il check per la vittoria o la sconfitta e in questi casi invocano le div appropriate, ci sono poi funzioni per resettare le barre ai valori iniziali.

MAIN.JS

Unisce molti dei file nel programma, ci sono presenti inizializzazioni per i mazzi, i cimiteri, lo shop, il mana (di fatto è l’energia), le mani, la board e vari pulsanti per finire il turno e per i casi di vittoria/sconfitta.

Come funzioni abbiamo:

iniziopartita che dà i valori necessari all’inizio della partita e crea gli oggetti utili come mazzi, shop, e fa pescare le carte.

fineTurno che resetta il mana, svuota il campo e le mani, per poi far ripescare una nuova mano dal mazzo, poi aggiorna la div che mostra il numero di carte nel cimitero, resetta il negozio, le armature l energia, aumenta il contatore dei turni e fa giocare l avversario.

Poi ci sono una serie di funzioni ausiliare allo shop che aggiornano e resettano i valori necessari per comprare le carte.

CreaCartaShop aggiorna la grafica dello shop, e genera randomicamente delle carte che compaiono nello shop e dei bottoni per comprare le carte

bottoneCompraAct fa i controlli necessari e passa le carte dallo shop al cimitero

shopToCimitero quando una carta viene comprata nello shop scala il costo aggiornando le div, e la carta viene passata al cimitero,

creaCartaDaNome passando il nome della carta restituisce la carta creata da carta.js

carteOnClick permette di mostrare il pulsante per giocare le carte dopo aver cliccato le stesse.

BottoneOnClick esegue un controllo se si ha abbastanza energia per giocare la carta, consuma l’energia, prende la carta e la sposta sul board togliendola dalla mano, nasconde il bottone per giocarla, aggiorna i valori per lo shop, e esegue l’effetto della carta in base al tipo della stessa.

giocatorePesca crea graficamente le carte e le pesca dal mazzo se è possibile

creaCartaGiocatore associa l’immagine corretta alla carta, gli dà un id, ci associa un bottone per poterla giocare.

avversarioPesca fa i controlli necessari e passa le carte dal mazzo avversario alla mano avversaria

creaCartaAvversario associa l’immagine corretta alla carta e gli dà un id

decideRetro randomizza un numero da zero a 8(numero di immagini per i retro che abbiamo)

avversarioGioca passa la carta data dalla mano avversaria alla board, e esegue l’effetto della carta in base al tipo

updateCarteInCimi aggiorna la div che mostra il numero di carte nel cimitero

myDumbPlay decide la carta da giocare cercando un modo per consumare tutto l energia a disposizione del avversario, fa questo creando degli array dove ci si inseriscono informazioni utili, e sui valori di costo messi negli array partendo sempre da una posizione diversa e percorrendo tutto l array (anche le posizioni precedenti a quella di partenza), fa un controllo se la carta è giocabile e in caso aggiorna il numero di energia rimanente e segna che quella carta è stata giocata, se l erray trovato ha energia rimanente minore di un array salvato aggiorna il salvato e continua il processo, poi torna le carte nel array trovato.

Successivamente ci sono delle funzioni per gestire i bottoni che compaiono dopo la vittoria/sconfitta.

InviaDataInscriversi

inviaDataAccedi

MENU.HTML

Mette il titolo al menu, crea i bottoni centrati e ha le funzioni necessarie per muovere l’utente alla schermata corretta a seconda del bottone premuto.

SHOP.JS

Come attributo ha un array che contiene il numero di carte diviso per tipo a disposizione per il giocatore, come funzioni ha:

cartaPerShop controlla se è stata richiesta una carta di un tipo specifico in caso passa il tipo corretto alla parte successiva sennò randomizza un numero e sceglie un tipo a seconda di quante carte dello stesso tipo il giocatore ha a disposizione, poi sceglie una carta in base al tipo selezionato

ci sono una serie di funzioni per aggiornare o ottenere i numeri nel array della classe

matriceIdCartePerTipo è una matrice che contiene tutte le carte divise per tipo

getRandomNumber sceglie un numero compreso tra un minimo e un massimo in maniera pseudo randomica

STILETAVOLA.CSS

Contiene le grafiche per gli elementi nella tavola

TAVOLA.HTML

Contiene gli elementi e le immagini necessarie per la tavola

CARTELLA IMMAGINI

Contiene le immagini per gli sfondi, le carte con la variante per lo shop e quella giocabile (tutto originale e creato da noi)

CARTELLA AiTester

Ha le cose necessarie per testare e migliorare l’intelligenza artificiale

TUTORIAL.HTML

Crea una pagina con quattro pulsanti e ti indirizza alla foto del tutorial giusta