

**ESCOLA ESTADUAL GOVERNADOR MILTON CAMPOS**

**Criado pelo Regulamento nº 27, de 05/02/1854 - Portaria nº 477/2001, de 13/06/2001**

**Rua Fernandes Tourinho, 1020 – Lourdes – Belo Horizonte - MG**

**CEP 30112-000 – Tel./fax (31) 2510-9779 – e-mail: escola.2135@educacao.mg.gov.br**

# TRABALHO JOGO DE CARTAS

Confronto elemental

CURSO – Desenvolvimento de Sistemas

PROFESSOR ORIENTADOR: Mizael

Souto

GRUPO: Gabriel Xavier e Gustavo Aquino

*JUNHO/ 2025*

**OBJETIVO DO JOGO:**

Ser o primeiro jogador a conquistar 10 pontos vencendo duelos mágicos entre cartas com elementos, forças e habilidades distintas.

**COMPONENTES**:

* 36 cartas exclusivas - Cada carta possui:
* Nome
* Tipo (Mago, Maga, Guardião, Conjurador, etc.)
* Elemento (Fogo, Água, Terra, Raio, Gelo, Luz, Lua, Vento, Espírito)
* Força (1 a 9)
* Habilidade especial
* Texto temático

**PREPARATIVOS**:

* 1. Embaralhe todas as 36 cartas.
  2. Cada jogador recebe 5 cartas.
  3. O restante forma o monte de compra, com uma pilha de descarte ao lado.

**TURNOS**:

* 1. Os dois jogadores escolhem secretamente 1 carta da mão.
  2. Revelam simultaneamente.
  3. Comparam a força das cartas:
* A maior força vence, a carta perdedora vai para o descarte.
* Em caso de empate, ambas são descartadas.

1. Aplica-se qualquer habilidade das cartas reveladas.
2. Quem vencer a rodada ganha 1 ponto.
3. Ambos compram nova carta, se houver no monte.
4. Repete-se até um jogador atingir 10 pontos.

**ELEMENTOS E VANTAGENS ELEMENTAIS:**

Elemento Vantagem sobre

Fogo Gelo, Terra

Água Fogo, Raio

Terra Raio, Vento

Raio Água, Luz

Gelo Vento, Terra

Luz Espírito, Lua

Lua Gelo, Fogo

Vento Luz, Lua

Espírito Água, Vento

**EXEMPLO DE RODADA:**

* Jogador 1 joga "Guardião do Rio" (força 2, água)
* Jogador 2 joga "Feiticeiro das Chamas" (força 3, fogo)
* Água tem vantagem sobre fogo → Guardião do Rio sobe para força 3.
* Empate! Ambas cartas são descartadas.
* Habilidade do Guardião do Rio: copia efeito da última carta (n/a nesse exemplo).

**FIM DE JOGO:**

* Vence quem fizer 10 pontos primeiro.
* Em caso de fim de monte sem vencedor, ganha quem tiver mais pontos.

**MODO AVANÇADO (OPCIONAL):**

* Cada jogador monta seu próprio baralho com 18 cartas.
* Uso estratégico de habilidades e tipos de elementos torna o jogo mais competitivo.

**Cartas do Jogo (36 totais):**

* 1. Guardião do Rio – Água – Força 2 – Copia o efeito da última carta jogada.
  2. Feiticeiro das Chamas – Fogo – Força 3 – Ganha +2 se o oponente for Terra.
  3. Maga da Geada – Gelo – Força 5 – Congela a próxima carta do oponente (não pode ser jogada na rodada seguinte).
  4. Conjurador do Trovão – Raio – Força 4 – Causa 1 de dano direto (ponto) mesmo se perder.
  5. Guardião da Floresta – Terra – Força 6 – Recupera uma carta descartada.
  6. Mago da Luz Serena – Luz – Força 3 – Anula a habilidade do oponente.
  7. Oráculo da Lua Crescente – Lua – Força 4 – Revela a próxima carta do oponente.
  8. Maga dos Ventos Leves – Vento – Força 2 – Troca de carta com o oponente antes da rodada.
  9. Conjurador Espiritual – Espírito – Força 5 – Revive uma carta descartada aleatória.
  10. Aprendiz do Vulcão – Fogo – Força 2 – Ganha +1 se perder o duelo.
  11. Discípula da Maré – Água – Força 3 – Oponente descarta uma carta aleatória da mão.
  12. Sentinela Rochoso – Terra – Força 4 – Ignora penalidade de elemento.
  13. Guardião das Estrelas – Luz – Força 5 – Concede +1 de força a outra carta da mão.
  14. Sábio do Vento Norte – Vento – Força 6 – Imune a efeitos de gelo.
  15. Mística do Eclipse – Lua – Força 3 – Em empates, ganha o ponto.
  16. Conjurador do Relâmpago – Raio – Força 7 – Se vencer, o oponente perde 2 cartas.
  17. Guardião da Névoa – Gelo – Força 4 – Reduz em -1 a força do oponente.
  18. Guardião do Espírito Antigo – Espírito – Força 6 – Não pode ser descartado.
  19. Feiticeira Ardente – Fogo – Força 4 – Queima a habilidade do oponente.
  20. Guardião do Lago Sereno – Água – Força 5 – Em caso de derrota, volta à mão.
  21. Maga dos Cristais – Gelo – Força 3 – Dobra sua força se oponente for Vento.
  22. Místico do Templo Solar – Luz – Força 7 – Aumenta a força de todas cartas Luz em 1.
  23. Druida das Pedras – Terra – Força 3 – Ganha +2 se o monte estiver com 10 ou menos cartas.
  24. Feiticeiro da Tempestade – Raio – Força 5 – Em empates, ambos perdem 1 ponto.
  25. Oráculo Lunar – Lua – Força 4 – Permite olhar 2 cartas do topo do monte.
  26. Guardião Celeste – Vento – Força 3 – Impede que oponente use habilidades.
  27. Evocador das Sombras – Espírito – Força 4 – Copia a força da carta oponente.
  28. Mago das Cinzas – Fogo – Força 6 – Se for derrotado, causa 2 de dano.
  29. Guardião Submerso – Água – Força 6 – Remove habilidade do oponente se for Raio.
  30. Conjurador das Montanhas – Terra – Força 7 – Não é afetado por vantagens elementais.
  31. Guardião da Aurora – Luz – Força 4 – Impede empate.
  32. Maga do Véu Lunar – Lua – Força 5 – Transforma sua carta em cópia da carta oponente.
  33. Maga dos Vendavais – Vento – Força 6 – Faz o oponente embaralhar a mão e descartar 1 carta.
  34. Sábio Espiritual – Espírito – Força 3 – Impede reviver cartas no turno seguinte.
  35. Feiticeiro Carmesim – Fogo – Força 5 – Aumenta sua força em +1 por cada carta Fogo descartada.
  36. Místico Abissal – Água – Força 7 – Se vencer, compra 2 cartas extras.

**NOTAS FINAIS:**

* Cada carta tem seu uso estratégico.
* As habilidades combinadas com os elementos e forças criam profundidade tática. - O jogador deve observar o descarte, prever jogadas e usar habilidades no tempo certo para vencer.

Explore os elementos, domine os feitiços e vença o Confronto Elemental!

Código anexado:

<!DOCTYPE html>

<html lang="pt-br">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Confronto Elemental - Aprendizagem Mágica</title>

<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.4.0/css/all.min.css">

<link href="https://fonts.googleapis.com/css2?family=Cinzel:wght@400;700&family=MedievalSharp&display=swap" rel="stylesheet">

<style>

:root {

--fire: #ff4d00;

--water: #00a8ff;

--earth: #b8860b;

--lightning: #ffeb3b;

--ice: #00d4ff;

--light: #ffffff;

--moon: #9c27b0;

--wind: #4caf50;

--spirit: #607d8b;

--gold: #ffd700;

--silver: #c0c0c0;

--bronze: #cd7f32;

--menu-bg: radial-gradient(circle at center, #0f0c29, #302b63, #24243e);

--menu-border: linear-gradient(45deg, #ff00cc, #3333ff);

--button-glow: 0 0 15px rgba(0, 234, 255, 0.7);

--text-glow: 0 0 10px rgba(0, 234, 255, 0.8);

--card-width: 180px;

--card-height: 250px;

--knowledge-blue: #1a73e8;

--correct-green: #34a853;

--incorrect-red: #ea4335;

}

\* {

box-sizing: border-box;

margin: 0;

padding: 0;

}

body {

margin: 0;

font-family: 'Cinzel', serif;

background: var(--menu-bg);

color: white;

display: flex;

flex-direction: column;

align-items: center;

min-height: 100vh;

overflow-x: hidden;

position: relative;

}

/\* Efeitos de partículas \*/

.particles {

position: fixed;

top: 0;

left: 0;

width: 100%;

height: 100%;

pointer-events: none;

z-index: -1;

}

.particle {

position: absolute;

background: rgba(0, 234, 255, 0.5);

border-radius: 50%;

pointer-events: none;

}

/\* Menu Principal \*/

.main-menu {

width: 100%;

max-width: 1200px;

padding: 20px;

display: flex;

flex-direction: column;

align-items: center;

position: relative;

margin-top: 30px;

}

.menu-header {

text-align: center;

margin-bottom: 40px;

position: relative;

width: 100%;

}

.game-title {

font-size: 5rem;

font-weight: 700;

color: var(--gold);

text-shadow: 0 0 20px var(--gold), 0 0 40px var(--gold), 0 0 60px var(--gold);

letter-spacing: 5px;

margin-bottom: 10px;

font-family: 'MedievalSharp', cursive;

position: relative;

z-index: 1;

animation: titleGlow 3s infinite alternate;

}

@keyframes titleGlow {

0% { text-shadow: 0 0 20px var(--gold), 0 0 40px var(--gold), 0 0 60px var(--gold); }

100% { text-shadow: 0 0 25px var(--gold), 0 0 50px var(--gold), 0 0 75px var(--gold); }

}

.game-subtitle {

font-size: 1.5rem;

color: var(--silver);

text-shadow: 0 0 10px var(--silver);

letter-spacing: 2px;

margin-bottom: 30px;

}

.menu-content {

display: flex;

flex-direction: column;

align-items: center;

width: 100%;

max-width: 800px;

background: rgba(0, 0, 0, 0.7);

border-radius: 20px;

padding: 40px;

box-shadow: 0 0 30px rgba(0, 234, 255, 0.3);

border: 2px solid;

border-image: var(--menu-border);

border-image-slice: 1;

position: relative;

overflow: hidden;

}

.menu-content::before {

content: "";

position: absolute;

top: -10px;

left: -10px;

right: -10px;

bottom: -10px;

background: linear-gradient(45deg,

rgba(255, 0, 204, 0.1),

rgba(51, 51, 255, 0.1),

rgba(255, 0, 204, 0.1));

z-index: -1;

border-radius: 25px;

animation: borderGlow 8s linear infinite;

}

@keyframes borderGlow {

0% { background-position: 0% 50%; }

50% { background-position: 100% 50%; }

100% { background-position: 0% 50%; }

}

.menu-buttons {

display: flex;

flex-direction: column;

gap: 25px;

width: 100%;

margin-bottom: 40px;

}

.menu-button {

position: relative;

background: linear-gradient(to right, #0f2027, #203a43, #2c5364);

color: white;

border: none;

padding: 18px 30px;

font-size: 1.5rem;

border-radius: 50px;

cursor: pointer;

box-shadow: var(--button-glow);

transition: all 0.3s ease;

font-family: 'Cinzel', serif;

letter-spacing: 1px;

text-align: center;

overflow: hidden;

z-index: 1;

width: 100%;

}

.menu-button::before {

content: "";

position: absolute;

top: 0;

left: -100%;

width: 100%;

height: 100%;

background: linear-gradient(90deg, transparent, rgba(0, 234, 255, 0.2), transparent);

transition: 0.5s;

z-index: -1;

}

.menu-button:hover {

transform: translateY(-5px) scale(1.03);

box-shadow: 0 0 25px rgba(0, 234, 255, 0.9);

}

.menu-button:hover::before {

left: 100%;

}

.menu-button i {

margin-right: 15px;

}

.difficulty-select {

display: flex;

justify-content: center;

gap: 20px;

margin: 30px 0;

flex-wrap: wrap;

}

.difficulty-button {

position: relative;

padding: 12px 25px;

font-size: 1.2rem;

background: rgba(0, 0, 0, 0.5);

color: #fff;

border: 2px solid #3498db;

border-radius: 50px;

cursor: pointer;

transition: all 0.3s;

font-family: 'Cinzel', serif;

min-width: 120px;

text-align: center;

overflow: hidden;

}

.difficulty-button::before {

content: "";

position: absolute;

top: 0;

left: 0;

width: 100%;

height: 100%;

background: linear-gradient(to right, rgba(0, 202, 255, 0.1), rgba(0, 234, 255, 0.3));

transform: translateX(-100%);

transition: all 0.3s;

z-index: -1;

}

.difficulty-button:hover::before {

transform: translateX(0);

}

.difficulty-button.active {

background: linear-gradient(to right, #00c6ff, #0072ff);

color: #00202e;

border-color: #00eaff;

box-shadow: var(--button-glow);

}

/\* Telas do jogo \*/

.game-container {

display: none;

width: 100%;

max-width: 1200px;

padding: 20px;

flex-direction: column;

align-items: center;

}

.game-header {

display: flex;

justify-content: space-between;

width: 100%;

margin-bottom: 20px;

background: rgba(0, 0, 0, 0.7);

padding: 15px;

border-radius: 10px;

border: 1px solid #00eaff;

box-shadow: 0 0 15px rgba(0, 234, 255, 0.3);

}

.player-info {

display: flex;

flex-direction: column;

align-items: center;

padding: 10px;

background: rgba(0, 0, 0, 0.5);

border-radius: 10px;

min-width: 150px;

}

.player-info.active {

border: 2px solid #00eaff;

box-shadow: 0 0 15px #00eaff;

}

.player-name {

font-weight: bold;

font-size: 1.2rem;

margin-bottom: 5px;

}

.player-score {

font-size: 1.5rem;

color: #00ffcc;

}

.player-hand {

font-size: 0.9rem;

color: #aaa;

}

.game-area {

display: flex;

flex-direction: column;

width: 100%;

gap: 20px;

}

.battle-area {

display: flex;

justify-content: space-around;

align-items: center;

margin: 20px 0;

flex-wrap: wrap;

min-height: 300px;

}

.card {

background: linear-gradient(to bottom, #141e30, #243b55);

border: 2px solid #00d9ff;

border-radius: 12px;

padding: 12px;

text-align: center;

width: var(--card-width);

min-height: var(--card-height);

box-shadow: 0 0 10px rgba(0, 217, 255, 0.4);

font-size: 1rem;

position: relative;

transition: transform 0.2s, box-shadow 0.3s;

display: flex;

flex-direction: column;

cursor: pointer;

margin: 10px;

}

.card.selected {

transform: translateY(-20px);

box-shadow: 0 0 20px rgba(0, 217, 255, 0.8);

border-color: #00ffcc;

}

.card.disabled {

opacity: 0.5;

cursor: not-allowed;

}

.card-name {

font-weight: bold;

margin-bottom: 10px;

color: var(--gold);

border-bottom: 1px solid rgba(0, 217, 255, 0.3);

padding-bottom: 5px;

}

.card-type {

font-style: italic;

font-size: 0.8rem;

margin-bottom: 5px;

color: #aaa;

}

.card-icon {

font-size: 2.5rem;

margin: 10px 0;

}

.card-power {

font-size: 1.5rem;

font-weight: bold;

margin: 10px 0;

color: #00ffcc;

}

.card-element {

display: inline-block;

padding: 3px 8px;

border-radius: 15px;

font-size: 0.8rem;

font-weight: bold;

margin-bottom: 10px;

}

.card-ability {

font-size: 0.8rem;

margin-top: auto;

padding: 8px;

background: rgba(0, 0, 0, 0.3);

border-radius: 8px;

border-left: 2px solid #00d9ff;

}

.card-knowledge {

position: absolute;

top: 5px;

right: 5px;

background: var(--knowledge-blue);

color: white;

width: 25px;

height: 25px;

border-radius: 50%;

display: flex;

align-items: center;

justify-content: center;

font-size: 0.7rem;

font-weight: bold;

}

.hand {

display: flex;

justify-content: center;

flex-wrap: wrap;

gap: 15px;

margin: 20px 0;

}

.game-log {

background: rgba(0, 0, 0, 0.7);

border: 1px solid #00eaff;

border-radius: 10px;

padding: 15px;

max-height: 150px;

overflow-y: auto;

font-size: 0.9rem;

margin: 20px 0;

}

.game-log p {

margin: 5px 0;

line-height: 1.4;

}

.game-controls {

display: flex;

justify-content: center;

gap: 20px;

margin-top: 20px;

}

.end-turn-btn, .back-btn {

padding: 12px 25px;

font-size: 1.1rem;

border: none;

border-radius: 50px;

cursor: pointer;

transition: all 0.3s;

font-family: 'Cinzel', serif;

display: flex;

align-items: center;

gap: 10px;

}

.end-turn-btn {

background: linear-gradient(to right, #00c6ff, #0072ff);

color: white;

box-shadow: 0 0 15px rgba(0, 198, 255, 0.5);

}

.end-turn-btn:hover {

transform: translateY(-3px);

box-shadow: 0 0 20px rgba(0, 198, 255, 0.8);

}

.back-btn {

background: rgba(255, 255, 255, 0.1);

color: white;

border: 1px solid #ff4d00;

}

.back-btn:hover {

background: rgba(255, 77, 0, 0.2);

}

/\* Element colors \*/

.fire { color: var(--fire); border-color: var(--fire); }

.water { color: var(--water); border-color: var(--water); }

.earth { color: var(--earth); border-color: var(--earth); }

.lightning { color: var(--lightning); border-color: var(--lightning); }

.ice { color: var(--ice); border-color: var(--ice); }

.light { color: var(--light); border-color: var(--light); }

.moon { color: var(--moon); border-color: var(--moon); }

.wind { color: var(--wind); border-color: var(--wind); }

.spirit { color: var(--spirit); border-color: var(--spirit); }

.fire-bg { background-color: rgba(255, 77, 0, 0.1); }

.water-bg { background-color: rgba(0, 168, 255, 0.1); }

.earth-bg { background-color: rgba(184, 134, 11, 0.1); }

.lightning-bg { background-color: rgba(255, 235, 59, 0.1); }

.ice-bg { background-color: rgba(0, 212, 255, 0.1); }

.light-bg { background-color: rgba(255, 255, 255, 0.1); }

.moon-bg { background-color: rgba(156, 39, 176, 0.1); }

.wind-bg { background-color: rgba(76, 175, 80, 0.1); }

.spirit-bg { background-color: rgba(96, 125, 139, 0.1); }

/\* Animações \*/

@keyframes cardPlay {

0% { transform: scale(1); }

50% { transform: scale(1.1); }

100% { transform: scale(1); }

}

.card-played {

animation: cardPlay 0.5s ease;

}

/\* Efeito de dano \*/

@keyframes cardDamage {

0% { transform: translateX(0); }

25% { transform: translateX(-10px); }

50% { transform: translateX(10px); }

75% { transform: translateX(-10px); }

100% { transform: translateX(0); }

}

.card-damage {

animation: cardDamage 0.3s ease;

}

/\* Efeito de vitória \*/

@keyframes cardVictory {

0% { transform: scale(1); }

50% { transform: scale(1.2); box-shadow: 0 0 30px gold; }

100% { transform: scale(1); }

}

.card-victory {

animation: cardVictory 0.5s ease;

}

/\* Efeito de aprendizado \*/

@keyframes knowledgeGlow {

0% { box-shadow: 0 0 5px var(--knowledge-blue); }

50% { box-shadow: 0 0 15px var(--knowledge-blue); }

100% { box-shadow: 0 0 5px var(--knowledge-blue); }

}

.knowledge-card {

animation: knowledgeGlow 2s infinite;

border: 2px solid var(--knowledge-blue);

}

/\* Estilo para mensagens de batalha \*/

.battle-message {

position: fixed;

top: 50%;

left: 50%;

transform: translate(-50%, -50%);

background: rgba(0, 0, 0, 0.8);

padding: 20px 40px;

border-radius: 10px;

border: 2px solid #00eaff;

box-shadow: 0 0 20px rgba(0, 234, 255, 0.5);

z-index: 100;

font-size: 1.5rem;

text-align: center;

opacity: 0;

transition: opacity 0.3s;

}

.battle-message.show {

opacity: 1;

}

/\* Efeitos elementais \*/

.elemental-effect {

position: fixed;

width: 100%;

height: 100%;

top: 0;

left: 0;

pointer-events: none;

z-index: 50;

opacity: 0;

transition: opacity 0.5s;

}

.elemental-effect.show {

opacity: 0.7;

}

/\* Tela de quiz \*/

.quiz-container {

display: none;

position: fixed;

top: 0;

left: 0;

width: 100%;

height: 100%;

background: rgba(0, 0, 0, 0.8);

z-index: 200;

display: flex;

justify-content: center;

align-items: center;

}

.quiz-box {

background: linear-gradient(to bottom, #141e30, #243b55);

border: 2px solid #00eaff;

border-radius: 15px;

padding: 30px;

width: 90%;

max-width: 600px;

box-shadow: 0 0 30px rgba(0, 234, 255, 0.5);

}

.quiz-question {

font-size: 1.3rem;

margin-bottom: 25px;

color: var(--gold);

text-align: center;

}

.quiz-options {

display: flex;

flex-direction: column;

gap: 15px;

}

.quiz-option {

background: rgba(0, 0, 0, 0.3);

border: 1px solid #00a8ff;

border-radius: 8px;

padding: 15px;

cursor: pointer;

transition: all 0.3s;

text-align: center;

}

.quiz-option:hover {

background: rgba(0, 168, 255, 0.2);

transform: translateX(5px);

}

.quiz-option.correct {

background: rgba(52, 168, 83, 0.3);

border-color: var(--correct-green);

}

.quiz-option.incorrect {

background: rgba(234, 67, 53, 0.3);

border-color: var(--incorrect-red);

}

.quiz-explanation {

margin-top: 20px;

padding: 15px;

background: rgba(0, 0, 0, 0.5);

border-radius: 8px;

border-left: 3px solid var(--knowledge-blue);

display: none;

}

.quiz-next {

margin-top: 20px;

padding: 10px 20px;

background: linear-gradient(to right, #00c6ff, #0072ff);

color: white;

border: none;

border-radius: 50px;

cursor: pointer;

display: none;

}

/\* Progresso educacional \*/

.knowledge-progress {

position: fixed;

top: 20px;

right: 20px;

background: rgba(0, 0, 0, 0.7);

border: 1px solid var(--knowledge-blue);

border-radius: 10px;

padding: 10px;

display: flex;

flex-direction: column;

align-items: center;

z-index: 10;

}

.progress-title {

font-size: 0.9rem;

margin-bottom: 5px;

color: var(--silver);

}

.progress-bar {

width: 150px;

height: 10px;

background: rgba(0, 0, 0, 0.5);

border-radius: 5px;

overflow: hidden;

margin-bottom: 5px;

}

.progress-fill {

height: 100%;

background: linear-gradient(to right, #00c6ff, #0072ff);

width: 0%;

transition: width 0.5s;

}

.progress-text {

font-size: 0.8rem;

color: var(--silver);

}

/\* Tela de recompensas \*/

.reward-container {

display: none;

position: fixed;

top: 0;

left: 0;

width: 100%;

height: 100%;

background: rgba(0, 0, 0, 0.9);

z-index: 200;

display: flex;

justify-content: center;

align-items: center;

flex-direction: column;

}

.reward-box {

background: linear-gradient(to bottom, #141e30, #243b55);

border: 2px solid var(--gold);

border-radius: 15px;

padding: 30px;

width: 90%;

max-width: 500px;

text-align: center;

box-shadow: 0 0 30px rgba(255, 215, 0, 0.5);

}

.reward-title {

font-size: 2rem;

color: var(--gold);

margin-bottom: 20px;

}

.reward-message {

font-size: 1.2rem;

margin-bottom: 30px;

}

.reward-card {

margin: 20px auto;

transform: scale(1.2);

}

.reward-button {

padding: 12px 30px;

background: linear-gradient(to right, #00c6ff, #0072ff);

color: white;

border: none;

border-radius: 50px;

cursor: pointer;

font-size: 1.1rem;

margin-top: 20px;

}

/\* Responsividade \*/

@media (max-width: 768px) {

.game-title {

font-size: 3rem;

}

.player-info {

min-width: 120px;

padding: 5px;

}

.player-name {

font-size: 1rem;

}

.player-score {

font-size: 1.2rem;

}

.card {

width: 140px;

min-height: 200px;

font-size: 0.9rem;

}

.card-icon {

font-size: 2rem;

}

.card-power {

font-size: 1.2rem;

}

.game-header {

flex-wrap: wrap;

gap: 10px;

}

}

@media (max-width: 480px) {

.game-title {

font-size: 2rem;

}

.game-subtitle {

font-size: 1rem;

}

.menu-button {

padding: 12px 20px;

font-size: 1.2rem;

}

.difficulty-button {

padding: 8px 15px;

font-size: 1rem;

min-width: 100px;

}

.card {

width: 120px;

min-height: 180px;

font-size: 0.8rem;

margin: 5px;

}

.knowledge-progress {

top: 10px;

right: 10px;

padding: 5px;

}

.progress-bar {

width: 100px;

}

}

</style>

</head>

<body>

<!-- Efeito de partículas -->

<div class="particles" id="particles"></div>

<!-- Efeitos elementais -->

<div class="elemental-effect" id="fireEffect" style="background: radial-gradient(circle, rgba(255,77,0,0.5) 0%, rgba(255,77,0,0) 70%);"></div>

<div class="elemental-effect" id="waterEffect" style="background: radial-gradient(circle, rgba(0,168,255,0.5) 0%, rgba(0,168,255,0) 70%);"></div>

<div class="elemental-effect" id="earthEffect" style="background: radial-gradient(circle, rgba(184,134,11,0.5) 0%, rgba(184,134,11,0) 70%);"></div>

<div class="elemental-effect" id="iceEffect" style="background: radial-gradient(circle, rgba(0,212,255,0.5) 0%, rgba(0,212,255,0) 70%);"></div>

<div class="elemental-effect" id="lightningEffect" style="background: radial-gradient(circle, rgba(255,235,59,0.5) 0%, rgba(255,235,59,0) 70%);"></div>

<!-- Mensagens de batalha -->

<div class="battle-message" id="battleMessage"></div>

<!-- Quiz Educacional -->

<div class="quiz-container" id="quizContainer">

<div class="quiz-box">

<div class="quiz-question" id="quizQuestion"></div>

<div class="quiz-options" id="quizOptions"></div>

<div class="quiz-explanation" id="quizExplanation"></div>

<button class="quiz-next" id="quizNext">Continuar</button>

</div>

</div>

<!-- Tela de Recompensas -->

<div class="reward-container" id="rewardContainer">

<div class="reward-box">

<h2 class="reward-title">Conhecimento Adquirido!</h2>

<p class="reward-message" id="rewardMessage"></p>

<div class="reward-card" id="rewardCard"></div>

<button class="reward-button" id="rewardButton">Continuar</button>

</div>

</div>

<!-- Progresso Educacional -->

<div class="knowledge-progress">

<div class="progress-title">Progresso do Aprendizado</div>

<div class="progress-bar">

<div class="progress-fill" id="knowledgeProgress"></div>

</div>

<div class="progress-text" id="knowledgeText">0/0 conceitos</div>

</div>

<!-- Menu Principal -->

<div class="main-menu" id="mainMenu">

<div class="menu-header">

<h1 class="game-title">CONFRONTO ELEMENTAL</h1>

<p class="game-subtitle">A Jornada do Aprendiz Mágico</p>

</div>

<div class="menu-content">

<div class="menu-buttons">

<button class="menu-button" id="startAdventureBtn">

<i class="fas fa-play"></i> Iniciar Aventura

</button>

<button class="menu-button" id="libraryBtn">

<i class="fas fa-book-open"></i> Biblioteca Arcana

</button>

<button class="menu-button" id="achievementsBtn">

<i class="fas fa-trophy"></i> Conquistas

</button>

<button class="menu-button" id="aboutBtn">

<i class="fas fa-info-circle"></i> Sobre o Jogo

</button>

</div>

<div class="difficulty-select">

<button class="difficulty-button active" id="easyBtn">

<i class="fas fa-feather-alt"></i> Aprendiz

</button>

<button class="difficulty-button" id="mediumBtn">

<i class="fas fa-hat-wizard"></i> Mago

</button>

<button class="difficulty-button" id="hardBtn">

<i class="fas fa-crown"></i> Arquimago

</button>

</div>

</div>

</div>

<!-- Tela de Jogo -->

<div class="game-container" id="gameScreen">

<div class="game-header">

<div class="player-info active" id="player1Info">

<div class="player-name">Aprendiz</div>

<div class="player-score">0</div>

<div class="player-hand">Cartas: 5</div>

</div>

<div class="player-info" id="player2Info">

<div class="player-name">Mestre Elemental</div>

<div class="player-score">0</div>

<div class="player-hand">Cartas: 5</div>

</div>

</div>

<div class="game-area">

<div class="battle-area" id="battleArea">

<!-- Cartas de batalha serão inseridas aqui -->

</div>

<div class="hand" id="playerHand">

<!-- Cartas da mão do jogador serão inseridas aqui -->

</div>

<div class="game-log" id="gameLog">

<!-- Registro do jogo será inserido aqui -->

</div>

<div class="game-controls">

<button class="end-turn-btn" id="playCardBtn">

<i class="fas fa-hand-paper"></i> Conjurar Carta

</button>

<button class="back-btn" id="backToMenuBtn">

<i class="fas fa-door-open"></i> Sair

</button>

</div>

</div>

</div>

<!-- Biblioteca Arcana -->

<div class="game-container" id="libraryScreen">

<div class="menu-header">

<h1 class="game-title">Biblioteca Arcana</h1>

<p class="game-subtitle">Conhecimento dos Elementos</p>

</div>

<div class="menu-content">

<div class="knowledge-tabs">

<button class="menu-button" id="elementsTabBtn">

<i class="fas fa-atom"></i> Elementos

</button>

<button class="menu-button" id="spellsTabBtn">

<i class="fas fa-scroll"></i> Feitiços

</button>

<button class="menu-button" id="progressTabBtn">

<i class="fas fa-medal"></i> Progresso

</button>

</div>

<div class="library-content" id="elementsTab">

<h2>Sistema Elemental</h2>

<p>O mundo mágico é governado por 8 elementos fundamentais, cada um com suas propriedades únicas:</p>

<div class="element-grid" id="elementGrid">

<!-- Os elementos serão inseridos aqui via JavaScript -->

</div>

</div>

<div class="library-content" id="spellsTab" style="display:none;">

<h2>Feitiços Desbloqueados</h2>

<p>Seu grimório pessoal contém todos os feitiços que você aprendeu em sua jornada:</p>

<div class="spells-grid" id="spellsGrid">

<!-- Os feitiços serão inseridos aqui via JavaScript -->

</div>

</div>

<div class="library-content" id="achievementsTab" style="display:none;">

<h2>Seu Progresso</h2>

<p>Registro de suas conquistas e conhecimentos adquiridos:</p>

<div class="achievements-list" id="achievementsList">

<!-- As conquistas serão inseridas aqui via JavaScript -->

</div>

</div>

<button class="menu-button" id="backFromLibraryBtn">

<i class="fas fa-arrow-left"></i> Voltar

</button>

</div>

</div>

<!-- Tela de Conquistas -->

<div class="game-container" id="achievementsScreen">

<div class="menu-header">

<h1 class="game-title">Conquistas</h1>

<p class="game-subtitle">Marcos de sua Jornada</p>

</div>

<div class="menu-content">

<div class="achievements-grid" id="achievementsGrid">

<!-- As conquistas serão inseridas aqui via JavaScript -->

</div>

<button class="menu-button" id="backFromAchievementsBtn">

<i class="fas fa-arrow-left"></i> Voltar

</button>

</div>

</div>

<!-- Tela Sobre o Jogo -->

<div class="game-container" id="aboutScreen">

<div class="menu-header">

<h1 class="game-title">Sobre o Jogo</h1>

<p class="game-subtitle">Confronto Elemental - A Jornada do Aprendiz Mágico</p>

</div>

<div class="menu-content">

<div class="about-content">

<p>Confronto Elemental é um jogo educacional que combina estratégia de cartas com aprendizagem de conceitos científicos através da magia elemental.</p>

<h2>Como Jogar</h2>

<p>• Cada carta representa um feitiço elemental com propriedades únicas</p>

<p>• Responda perguntas corretamente para fortalecer seus feitiços</p>

<p>• Aprenda sobre ciência através do sistema de elementos e suas interações</p>

<h2>Conceitos Educacionais</h2>

<p>O jogo ensina princípios de:</p>

<p>• Química (estados da matéria, reações químicas)</p>

<p>• Física (eletricidade, termodinâmica)</p>

<p>• Ciências ambientais (ciclo da água, erosão)</p>

<p>• Astronomia (fases da lua, ventos solares)</p>

<h2>Créditos</h2>

<p>Desenvolvido para fins educacionais por entusiastas de jogos e educação.</p>

</div>

<button class="menu-button" id="backFromAboutBtn">

<i class="fas fa-arrow-left"></i> Voltar

</button>

</div>

</div>

<script>

// Configurações do jogo com foco educacional

const gameConfig = {

difficulty: 'easy',

playerDeck: [],

maxPoints: 5,

currentPlayer: 0,

selectedCard: null,

gameState: 'waiting',

players: [

{ id: 0, name: "Aprendiz", score: 0, hand: [], deck: [], isHuman: true, knowledge: 0 },

{ id: 1, name: "Mestre Elemental", score: 0, hand: [], deck: [], isHuman: false }

],

deck: [],

discardPile: [],

gameLog: [],

round: 0,

cardsPlayed: 0,

currentSubject: 'chemistry', // Matéria atual (chemistry, physics, etc.)

knowledgePoints: 0, // Pontos de conhecimento acumulados

totalQuestions: 0, // Total de perguntas respondidas

correctAnswers: 0, // Respostas corretas

unlockedCards: [], // Cartas desbloqueadas

achievements: [], // Conquistas obtidas

currentQuiz: null, // Quiz atual

elementsMastery: { // Domínio dos elementos

fire: 0,

water: 0,

earth: 0,

air: 0,

ice: 0,

lightning: 0,

moon: 0,

spirit: 0

}

};

// Vantagens elementais baseadas em ciência real

const elementAdvantages = {

fire: ['ice', 'spirit'], // Fogo derrete gelo, purifica espíritos

water: ['fire', 'earth'], // Água apaga fogo, erode terra

earth: ['lightning', 'water'], // Terra conduz eletricidade, absorve água

lightning: ['water', 'air'], // Eletricidade conduz na água, ioniza ar

ice: ['earth', 'water'], // Gelo quebra terra, congela água

air: ['fire', 'lightning'], // Ar alimenta fogo, dissipa eletricidade

moon: ['water', 'spirit'], // Lua controla marés, influencia emoções

spirit: ['earth', 'moon'] // Espíritos transcendem matéria, influenciados pela lua

};

// Banco de perguntas educacionais organizado por assunto e elemento

const questionBank = {

chemistry: [

{

question: "Qual elemento é essencial para a combustão?",

options: ["Oxigênio", "Hidrogênio", "Nitrogênio", "Dióxido de Carbono"],

answer: 0,

explanation: "O oxigênio é necessário para a maioria das reações de combustão, atuando como comburente.",

element: "fire",

difficulty: 1

},

{

question: "O que acontece quando um ácido reage com uma base?",

options: ["Forma-se um sal e água", "Ocorre combustão", "Libera hidrogênio", "Cria um metal"],

answer: 0,

explanation: "Ácidos e bases neutralizam-se mutuamente em uma reação de neutralização, formando sal e água.",

element: "water",

difficulty: 2

}

],

physics: [

{

question: "O que é necessário para que ocorra condução elétrica?",

options: ["Elétrons livres", "Átomos estáveis", "Temperatura baixa", "Pressão alta"],

answer: 0,

explanation: "A condução elétrica ocorre quando há elétrons livres que podem se mover através do material.",

element: "lightning",

difficulty: 1

},

{

question: "Qual fenômeno explica por que a água esquenta mais devagar que a terra?",

options: ["Calor específico", "Condutividade térmica", "Expansão térmica", "Radiação infravermelha"],

answer: 0,

explanation: "A água tem alto calor específico, o que significa que precisa de mais energia para aumentar sua temperatura.",

element: "earth",

difficulty: 2

}

],

// Adicione mais matérias e perguntas conforme necessário

};

// Todas as cartas do jogo com temas educacionais

const allCards = [

// Cartas básicas (nível iniciante)

{

id: 1,

name: "Bola de Fogo",

type: "Feitiço Elemental",

element: "fire",

power: 3,

ability: "Ensina sobre combustão e reações exotérmicas.",

tier: 1,

icon: "🔥",

subject: "chemistry",

questionId: 0 // Índice da pergunta relacionada

},

{

id: 2,

name: "Jato d'Água",

type: "Feitiço Elemental",

element: "water",

power: 2,

ability: "Introduz conceitos de densidade e pressão hidrostática.",

tier: 1,

icon: "💧",

subject: "physics",

questionId: 0

},

{

id: 3,

name: "Barreira Terrea",

type: "Encantamento",

element: "earth",

power: 4,

ability: "Explica a formação de rochas e minerais.",

tier: 1,

icon: "🌍",

subject: "geology",

questionId: 0

},

// Cartas intermediárias (desbloqueadas com progresso)

{

id: 21,

name: "Meteoro Flamejante",

type: "Feitiço Avançado",

element: "fire",

power: 5,

ability: "Demonstra a energia cinética e térmica de meteoros.",

tier: 2,

icon: "☄️",

subject: "astronomy",

questionId: 0,

unlockRequirement: { knowledge: 10, element: "fire" }

},

// Cartas avançadas

{

id: 31,

name: "Fênix Renascida",

type: "Criatura Lendária",

element: "fire",

power: 7,

ability: "Ilustra o ciclo de vida estelar e renascimento cósmico.",

tier: 3,

icon: "🦚",

subject: "astronomy",

questionId: 0,

unlockRequirement: { knowledge: 25, element: "fire" }

},

// Adicione mais cartas conforme necessário

];

// Conquistas educacionais

const allAchievements = [

{

id: 1,

name: "Aprendiz do Fogo",

description: "Responda corretamente 5 perguntas sobre combustão",

icon: "🔥",

completed: false,

reward: { cardId: 21, knowledge: 5 }

},

{

id: 2,

name: "Mestre da Água",

description: "Domine todos os conceitos do elemento água",

icon: "💧",

completed: false,

reward: { cardId: 22, knowledge: 10 }

},

// Adicione mais conquistas

];

// Inicializa o jogo quando a página carrega

document.addEventListener('DOMContentLoaded', () => {

initGame();

setupEventListeners();

});

// Configura todos os event listeners

function setupEventListeners() {

// Botões do menu principal

document.getElementById('startAdventureBtn').addEventListener('click', startAdventure);

document.getElementById('libraryBtn').addEventListener('click', () => showScreen('libraryScreen'));

document.getElementById('achievementsBtn').addEventListener('click', () => showScreen('achievementsScreen'));

document.getElementById('aboutBtn').addEventListener('click', () => showScreen('aboutScreen'));

// Botões de dificuldade

document.getElementById('easyBtn').addEventListener('click', () => setDifficulty('easy'));

document.getElementById('mediumBtn').addEventListener('click', () => setDifficulty('medium'));

document.getElementById('hardBtn').addEventListener('click', () => setDifficulty('hard'));

// Botões de jogo

document.getElementById('playCardBtn').addEventListener('click', playSelectedCard);

document.getElementById('backToMenuBtn').addEventListener('click', backToMenu);

// Botões da biblioteca

document.getElementById('elementsTabBtn').addEventListener('click', () => showLibraryTab('elements'));

document.getElementById('spellsTabBtn').addEventListener('click', () => showLibraryTab('spells'));

document.getElementById('progressTabBtn').addEventListener('click', () => showLibraryTab('achievements'));

document.getElementById('backFromLibraryBtn').addEventListener('click', backToMenu);

// Botões de conquistas

document.getElementById('backFromAchievementsBtn').addEventListener('click', backToMenu);

// Botões sobre o jogo

document.getElementById('backFromAboutBtn').addEventListener('click', backToMenu);

// Botão de próximo no quiz

document.getElementById('quizNext').addEventListener('click', handleQuizNext);

// Botão de recompensa

document.getElementById('rewardButton').addEventListener('click', handleRewardContinue);

}

// Inicializa o jogo

function initGame() {

createParticles();

loadPlayerProgress();

updateKnowledgeProgress();

// Configura decks iniciais

setupDecks();

// Mostra o menu principal

showScreen('mainMenu');

}

// Cria partículas decorativas

function createParticles() {

const particlesContainer = document.getElementById('particles');

const particleCount = Math.floor(window.innerWidth / 10);

for (let i = 0; i < particleCount; i++) {

const particle = document.createElement('div');

particle.classList.add('particle');

// Posição aleatória

const posX = Math.random() \* window.innerWidth;

const posY = Math.random() \* window.innerHeight;

// Tamanho aleatório

const size = Math.random() \* 3 + 1;

// Duração da animação aleatória

const duration = Math.random() \* 20 + 10;

// Atraso aleatório

const delay = Math.random() \* 20;

particle.style.width = `${size}px`;

particle.style.height = `${size}px`;

particle.style.left = `${posX}px`;

particle.style.top = `${posY}px`;

particle.style.animation = `float ${duration}s linear ${delay}s infinite`;

particlesContainer.appendChild(particle);

}

}

// Carrega o progresso do jogador do localStorage

function loadPlayerProgress() {

const savedProgress = localStorage.getItem('elementalConfrontationProgress');

if (savedProgress) {

const progress = JSON.parse(savedProgress);

gameConfig.knowledgePoints = progress.knowledgePoints || 0;

gameConfig.totalQuestions = progress.totalQuestions || 0;

gameConfig.correctAnswers = progress.correctAnswers || 0;

gameConfig.unlockedCards = progress.unlockedCards || [];

gameConfig.achievements = progress.achievements || allAchievements.map(a => ({...a, completed: false}));

gameConfig.elementsMastery = progress.elementsMastery || {...gameConfig.elementsMastery};

// Verifica conquistas completas

checkAchievements();

} else {

// Progresso inicial

gameConfig.achievements = [...allAchievements];

gameConfig.unlockedCards = allCards.filter(card => card.tier === 1);

}

}

// Salva o progresso do jogador no localStorage

function savePlayerProgress() {

const progress = {

knowledgePoints: gameConfig.knowledgePoints,

totalQuestions: gameConfig.totalQuestions,

correctAnswers: gameConfig.correctAnswers,

unlockedCards: gameConfig.unlockedCards,

achievements: gameConfig.achievements,

elementsMastery: gameConfig.elementsMastery

};

localStorage.setItem('elementalConfrontationProgress', JSON.stringify(progress));

}

// Atualiza a barra de progresso do conhecimento

function updateKnowledgeProgress() {

const totalPossible = Object.keys(questionBank).reduce((sum, subject) => sum + questionBank[subject].length, 0);

const progress = (gameConfig.correctAnswers / totalPossible) \* 100;

document.getElementById('knowledgeProgress').style.width = `${progress}%`;

document.getElementById('knowledgeText').textContent =

`${gameConfig.correctAnswers}/${totalPossible} conceitos`;

}

// Configura os decks dos jogadores

function setupDecks() {

// Deck do jogador - começa com cartas básicas

gameConfig.players[0].deck = [...gameConfig.unlockedCards];

// Deck do oponente - cartas baseadas na dificuldade

const opponentCards = allCards.filter(card =>

card.tier <= (gameConfig.difficulty === 'easy' ? 1 : gameConfig.difficulty === 'medium' ? 2 : 3)

);

gameConfig.players[1].deck = [...opponentCards];

// Embaralha os decks

shuffleDecks();

// Distribui 5 cartas para cada jogador

dealInitialHands();

}

// Embaralha os decks

function shuffleDecks() {

gameConfig.players.forEach(player => {

player.deck = shuffleDeck(player.deck);

});

}

// Função para embaralhar um deck

function shuffleDeck(deck) {

const newDeck = [...deck];

for (let i = newDeck.length - 1; i > 0; i--) {

const j = Math.floor(Math.random() \* (i + 1));

[newDeck[i], newDeck[j]] = [newDeck[j], newDeck[i]];

}

return newDeck;

}

// Distribui as mãos iniciais

function dealInitialHands() {

gameConfig.players.forEach(player => {

player.hand = player.deck.splice(0, 5);

});

}

// Inicia uma nova aventura

function startAdventure() {

setupDecks();

showScreen('gameScreen');

updatePlayerInfo();

renderPlayerHand();

// Primeiro turno do jogador

gameConfig.gameState = 'waiting';

gameConfig.currentPlayer = 0;

updateGameLog("Bem-vindo ao Torneio Elemental! Seu turno começa.");

}

// Atualiza as informações dos jogadores na tela

function updatePlayerInfo() {

gameConfig.players.forEach((player, index) => {

const playerElement = document.getElementById(`player${index + 1}Info`);

if (playerElement) {

playerElement.querySelector('.player-name').textContent = player.name;

playerElement.querySelector('.player-score').textContent = player.score;

playerElement.querySelector('.player-hand').textContent = `Cartas: ${player.hand.length}`;

// Destaca o jogador atual

if (index === gameConfig.currentPlayer) {

playerElement.classList.add('active');

} else {

playerElement.classList.remove('active');

}

}

});

}

// Renderiza a mão do jogador

function renderPlayerHand() {

const handElement = document.getElementById('playerHand');

handElement.innerHTML = '';

gameConfig.players[0].hand.forEach((card, index) => {

const cardElement = createCardElement(card, index);

handElement.appendChild(cardElement);

});

}

// Cria um elemento HTML para uma carta

function createCardElement(card, index) {

const cardElement = document.createElement('div');

cardElement.className = `card ${card.element}`;

cardElement.dataset.index = index;

// Adiciona classe de conhecimento se o jogador respondeu perguntas sobre este elemento

if (gameConfig.elementsMastery[card.element] > 0) {

cardElement.classList.add('knowledge-card');

}

// Conteúdo da carta

cardElement.innerHTML = `

<div class="card-name">${card.name}</div>

<div class="card-type">${card.type}</div>

<div class="card-icon">${card.icon}</div>

<div class="card-power">Força: ${card.power}</div>

<div class="card-element ${card.element}">${getElementName(card.element)}</div>

<div class="card-ability">${card.ability}</div>

${gameConfig.elementsMastery[card.element] > 0 ?

`<div class="card-knowledge">${gameConfig.elementsMastery[card.element]}</div>` : ''}

`;

// Adiciona evento de clique

cardElement.addEventListener('click', () => selectCard(index));

return cardElement;

}

// Retorna o nome do elemento formatado

function getElementName(element) {

const elementNames = {

fire: 'Fogo',

water: 'Água',

earth: 'Terra',

lightning: 'Raio',

ice: 'Gelo',

air: 'Ar',

moon: 'Lua',

spirit: 'Espírito'

};

return elementNames[element] || element;

}

// Seleciona uma carta para jogar

function selectCard(index) {

// Limpa seleção anterior

const cards = document.querySelectorAll('.card');

cards.forEach(card => card.classList.remove('selected'));

// Seleciona nova carta

cards[index].classList.add('selected');

gameConfig.selectedCard = index;

updateGameLog(`Carta selecionada: ${gameConfig.players[0].hand[index].name}`);

}

// Joga a carta selecionada

function playSelectedCard() {

if (gameConfig.selectedCard === null) {

updateGameLog("Selecione uma carta para jogar!");

return;

}

const selectedCard = gameConfig.players[0].hand[gameConfig.selectedCard];

// Verifica se há perguntas relacionadas a esta carta

const cardQuestions = questionBank[selectedCard.subject]?.filter(q => q.element === selectedCard.element);

if (cardQuestions && cardQuestions.length > 0) {

// Mostra quiz antes de jogar a carta

showQuiz(selectedCard);

} else {

// Joga a carta diretamente se não houver perguntas

playCard(selectedCard);

}

}

// Mostra o quiz relacionado à carta

function showQuiz(card) {

gameConfig.gameState = 'quiz';

// Encontra perguntas relacionadas ao elemento da carta

const questions = questionBank[card.subject].filter(q => q.element === card.element);

const randomQuestion = questions[Math.floor(Math.random() \* questions.length)];

// Configura o quiz

gameConfig.currentQuiz = {

card: card,

question: randomQuestion,

answered: false

};

// Mostra a pergunta

document.getElementById('quizQuestion').textContent = randomQuestion.question;

// Mostra as opções

const optionsContainer = document.getElementById('quizOptions');

optionsContainer.innerHTML = '';

randomQuestion.options.forEach((option, index) => {

const optionElement = document.createElement('div');

optionElement.className = 'quiz-option';

optionElement.textContent = option;

optionElement.dataset.index = index;

optionElement.addEventListener('click', () => selectAnswer(index));

optionsContainer.appendChild(optionElement);

});

// Esconde explicação e botão próximo

document.getElementById('quizExplanation').style.display = 'none';

document.getElementById('quizNext').style.display = 'none';

// Mostra o container do quiz

document.getElementById('quizContainer').style.display = 'flex';

}

// Seleciona uma resposta no quiz

function selectAnswer(selectedIndex) {

if (gameConfig.currentQuiz.answered) return;

gameConfig.currentQuiz.answered = true;

gameConfig.totalQuestions++;

const question = gameConfig.currentQuiz.question;

const options = document.querySelectorAll('.quiz-option');

// Marca a resposta correta

options[question.answer].classList.add('correct');

// Verifica se a resposta está correta

if (selectedIndex === question.answer) {

gameConfig.correctAnswers++;

gameConfig.knowledgePoints += question.difficulty;

gameConfig.elementsMastery[gameConfig.currentQuiz.card.element]++;

// Bônus para carta

gameConfig.currentQuiz.card.power += question.difficulty;

updateGameLog(`Resposta correta! ${gameConfig.currentQuiz.card.name} ganhou +${question.difficulty} de força!`);

} else {

options[selectedIndex].classList.add('incorrect');

updateGameLog("Resposta incorreta. Revise o conceito e tente novamente.");

}

// Mostra explicação

const explanationElement = document.getElementById('quizExplanation');

explanationElement.textContent = question.explanation;

explanationElement.style.display = 'block';

// Mostra botão para continuar

document.getElementById('quizNext').style.display = 'block';

// Atualiza progresso

updateKnowledgeProgress();

checkAchievements();

savePlayerProgress();

}

// Manipula o clique no botão "Continuar" do quiz

function handleQuizNext() {

document.getElementById('quizContainer').style.display = 'none';

// Se houver uma carta associada ao quiz, jogue-a

if (gameConfig.currentQuiz && gameConfig.currentQuiz.card) {

playCard(gameConfig.currentQuiz.card);

}

// Limpa o quiz atual

gameConfig.currentQuiz = null;

}

// Joga uma carta no campo de batalha

function playCard(card) {

if (gameConfig.gameState !== 'waiting') return;

gameConfig.gameState = 'playing';

// Remove a carta da mão do jogador

const cardIndex = gameConfig.players[0].hand.findIndex(c => c.id === card.id);

if (cardIndex !== -1) {

gameConfig.players[0].hand.splice(cardIndex, 1);

}

// Adiciona ao descarte

gameConfig.discardPile.push(card);

// Atualiza a interface

updatePlayerInfo();

renderPlayerHand();

// Mostra a carta jogada

const battleArea = document.getElementById('battleArea');

battleArea.innerHTML = '';

const playerCardElement = createCardElement(card);

playerCardElement.classList.add('card-played');

battleArea.appendChild(playerCardElement);

// Oponente joga uma carta

setTimeout(() => {

playOpponentCard(card);

}, 1000);

}

// Oponente joga uma carta

function playOpponentCard(playerCard) {

const opponent = gameConfig.players[1];

// Escolhe uma carta aleatória da mão do oponente

const randomIndex = Math.floor(Math.random() \* opponent.hand.length);

const opponentCard = opponent.hand[randomIndex];

// Remove a carta da mão do oponente

opponent.hand.splice(randomIndex, 1);

// Mostra a carta do oponente

const battleArea = document.getElementById('battleArea');

const opponentCardElement = createCardElement(opponentCard);

opponentCardElement.classList.add('card-played');

battleArea.appendChild(opponentCardElement);

// Processa os resultados da batalha

setTimeout(() => {

resolveBattle(playerCard, opponentCard);

}, 1000);

}

// Resolve a batalha entre duas cartas

function resolveBattle(playerCard, opponentCard) {

// Calcula o poder considerando vantagens elementais

let playerPower = playerCard.power;

let opponentPower = opponentCard.power;

// Verifica vantagens elementais

if (elementAdvantages[playerCard.element].includes(opponentCard.element)) {

playerPower += 2;

updateGameLog(`Vantagem elemental! ${playerCard.name} é forte contra ${opponentCard.name}`);

}

if (elementAdvantages[opponentCard.element].includes(playerCard.element)) {

opponentPower += 2;

updateGameLog(`Desvantagem elemental! ${opponentCard.name} é forte contra ${playerCard.name}`);

}

// Determina o vencedor

if (playerPower > opponentPower) {

// Jogador vence

gameConfig.players[0].score++;

updateGameLog(`Você venceu com ${playerCard.name} (${playerPower} vs ${opponentPower})!`);

// Animação de vitória

document.querySelectorAll('.card').forEach(card => {

if (card.textContent.includes(playerCard.name)) {

card.classList.add('card-victory');

}

});

// Aplica habilidade da carta vencedora

applyCardEffect(playerCard, opponentCard, true);

} else if (opponentPower > playerPower) {

// Oponente vence

gameConfig.players[1].score++;

updateGameLog(`${opponent.name} venceu com ${opponentCard.name} (${opponentPower} vs ${playerPower})!`);

// Aplica habilidade da carta vencedora

applyCardEffect(opponentCard, playerCard, false);

} else {

// Empate

updateGameLog(`Empate! ${playerCard.name} e ${opponentCard.name} se anulam.`);

}

// Verifica se o jogo terminou

checkGameEnd();

// Prepara próximo turno

prepareNextTurn();

}

// Aplica efeitos especiais da carta

function applyCardEffect(winnerCard, loserCard, isPlayerWinner) {

const winner = isPlayerWinner ? gameConfig.players[0] : gameConfig.players[1];

const loser = isPlayerWinner ? gameConfig.players[1] : gameConfig.players[0];

// Efeito baseado no ID da carta (poderia ser mais genérico com um sistema de habilidades)

switch(winnerCard.id) {

case 1: // Bola de Fogo

if (loserCard.element === 'ice') {

winner.score++;

updateGameLog(`Efeito especial: ${winnerCard.name} derrete ${loserCard.name}! Ponto extra.`);

}

break;

case 21: // Meteoro Flamejante

// Queima uma carta aleatória do oponente

if (loser.hand.length > 0) {

const randomIndex = Math.floor(Math.random() \* loser.hand.length);

const burnedCard = loser.hand.splice(randomIndex, 1)[0];

updateGameLog(`Efeito especial: ${winnerCard.name} queimou ${burnedCard.name} do oponente!`);

}

break;

// Adicione mais efeitos especiais aqui

}

}

// Prepara o próximo turno

function prepareNextTurn() {

// Limpa o campo de batalha após um delay

setTimeout(() => {

document.getElementById('battleArea').innerHTML = '';

// Cada jogador compra uma carta

drawCards();

// Verifica se alguém ganhou

if (gameConfig.players[0].score >= gameConfig.maxPoints) {

showReward("Você venceu o duelo!", "Ganhou 5 pontos de conhecimento por sua vitória!");

gameConfig.knowledgePoints += 5;

savePlayerProgress();

} else if (gameConfig.players[1].score >= gameConfig.maxPoints) {

showReward("Você perdeu o duelo.", "Estude mais e tente novamente!");

} else {

// Próximo turno do jogador

gameConfig.gameState = 'waiting';

gameConfig.currentPlayer = 0;

gameConfig.selectedCard = null;

updatePlayerInfo();

renderPlayerHand();

updateGameLog("Seu turno. Selecione uma carta para jogar.");

}

}, 2000);

}

// Mostra tela de recompensa

function showReward(title, message) {

document.getElementById('rewardMessage').textContent = message;

document.getElementById('rewardContainer').style.display = 'flex';

}

// Manipula o clique no botão "Continuar" da tela de recompensa

function handleRewardContinue() {

document.getElementById('rewardContainer').style.display = 'none';

backToMenu();

}

// Verifica se o jogo terminou

function checkGameEnd() {

if (gameConfig.players[0].score >= gameConfig.maxPoints) {

updateGameLog("Você venceu o duelo! Parabéns!");

return true;

} else if (gameConfig.players[1].score >= gameConfig.maxPoints) {

updateGameLog("Você perdeu o duelo. Tente novamente!");

return true;

}

return false;

}

// Compra cartas para os jogadores

function drawCards() {

gameConfig.players.forEach(player => {

if (player.deck.length > 0 && player.hand.length < 5) {

player.hand.push(player.deck.shift());

} else if (player.deck.length === 0 && player.hand.length === 0) {

// Fim do jogo - sem cartas

updateGameLog(`${player.name} ficou sem cartas!`);

}

});

updatePlayerInfo();

}

// Atualiza o registro do jogo

function updateGameLog(message) {

const logElement = document.getElementById('gameLog');

const entry = document.createElement('p');

entry.textContent = message;

logElement.appendChild(entry);

logElement.scrollTop = logElement.scrollHeight;

}

// Verifica conquistas

function checkAchievements() {

gameConfig.achievements.forEach(achievement => {

if (!achievement.completed) {

// Verifica condições para cada conquista

switch(achievement.id) {

case 1: // Aprendiz do Fogo

if (gameConfig.elementsMastery.fire >= 5) {

completeAchievement(achievement);

}

break;

case 2: // Mestre da Água

if (gameConfig.elementsMastery.water >= 10) {

completeAchievement(achievement);

}

break;

// Adicione mais condições para outras conquistas

}

}

});

}

// Completa uma conquista

function completeAchievement(achievement) {

achievement.completed = true;

// Adiciona recompensa

if (achievement.reward.cardId) {

const rewardCard = allCards.find(card => card.id === achievement.reward.cardId);

if (rewardCard && !gameConfig.unlockedCards.some(c => c.id === rewardCard.id)) {

gameConfig.unlockedCards.push(rewardCard);

updateGameLog(`Nova carta desbloqueada: ${rewardCard.name}!`);

}

}

if (achievement.reward.knowledge) {

gameConfig.knowledgePoints += achievement.reward.knowledge;

}

// Mostra notificação

showReward(`Conquista: ${achievement.name}`, achievement.description);

// Salva progresso

savePlayerProgress();

}

// Mostra uma tela específica

function showScreen(screenId) {

// Esconde todas as telas

document.querySelectorAll('.game-container').forEach(screen => {

screen.style.display = 'none';

});

// Esconde telas modais também

document.getElementById('quizContainer').style.display = 'none';

document.getElementById('rewardContainer').style.display = 'none';

// Mostra a tela solicitada

document.getElementById(screenId).style.display = 'flex';

// Configurações específicas para cada tela

if (screenId === 'libraryScreen') {

showLibraryTab('elements');

} else if (screenId === 'achievementsScreen') {

renderAchievements();

}

}

// Mostra uma aba específica na biblioteca

function showLibraryTab(tabId) {

// Esconde todas as abas

document.querySelectorAll('.library-content').forEach(tab => {

tab.style.display = 'none';

});

// Mostra a aba solicitada

document.getElementById(`${tabId}Tab`).style.display = 'block';

// Renderiza conteúdo específico

if (tabId === 'elements') {

renderElements();

} else if (tabId === 'spells') {

renderSpells();

} else if (tabId === 'achievements') {

renderLibraryAchievements();

}

}

// Renderiza os elementos na biblioteca

function renderElements() {

const elementGrid = document.querySelector('#elementsTab .element-grid');

elementGrid.innerHTML = '';

const elements = [

{ name: 'Fogo', icon: '🔥', description: 'Representa combustão, energia térmica e reações exotérmicas.', mastery: gameConfig.elementsMastery.fire },

{ name: 'Água', icon: '💧', description: 'Representa estados da matéria, soluções químicas e pressão hidrostática.', mastery: gameConfig.elementsMastery.water },

{ name: 'Terra', icon: '🌍', description: 'Representa geologia, minerais e a crosta terrestre.', mastery: gameConfig.elementsMastery.earth },

{ name: 'Raio', icon: '⚡', description: 'Representa eletricidade, magnetismo e energia elétrica.', mastery: gameConfig.elementsMastery.lightning },

{ name: 'Gelo', icon: '❄️', description: 'Representa estados da matéria, transições de fase e termodinâmica.', mastery: gameConfig.elementsMastery.ice },

{ name: 'Ar', icon: '🌬️', description: 'Representa atmosfera, pressão do ar e gases.', mastery: gameConfig.elementsMastery.air },

{ name: 'Lua', icon: '🌕', description: 'Representa gravitação, marés e ciclos naturais.', mastery: gameConfig.elementsMastery.moon },

{ name: 'Espírito', icon: '👻', description: 'Representa energia, consciência e fenômenos não-materiais.', mastery: gameConfig.elementsMastery.spirit }

];

elements.forEach(element => {

const elementCard = document.createElement('div');

elementCard.className = 'card';

elementCard.innerHTML = `

<div class="card-name">${element.name}</div>

<div class="card-icon">${element.icon}</div>

<div class="card-ability">${element.description}</div>

<div class="card-power">Domínio: ${element.mastery}</div>

`;

elementGrid.appendChild(elementCard);

});

}

// Renderiza os feitiços na biblioteca

function renderSpells() {

const spellsGrid = document.querySelector('#spellsTab .spells-grid');

spellsGrid.innerHTML = '';

// Agrupa cartas por elemento

const spellsByElement = {};

gameConfig.unlockedCards.forEach(card => {

if (!spellsByElement[card.element]) {

spellsByElement[card.element] = [];

}

spellsByElement[card.element].push(card);

});

// Cria seções para cada elemento

for (const element in spellsByElement) {

const elementSection = document.createElement('div');

elementSection.className = 'element-section';

elementSection.innerHTML = `<h3>${getElementName(element)}</h3>`;

const spellsContainer = document.createElement('div');

spellsContainer.className = 'spells-container';

spellsByElement[element].forEach(card => {

const spellCard = createCardElement(card);

spellsContainer.appendChild(spellCard);

});

elementSection.appendChild(spellsContainer);

spellsGrid.appendChild(elementSection);

}

}

// Renderiza as conquistas na biblioteca

function renderLibraryAchievements() {

const achievementsList = document.querySelector('#achievementsTab .achievements-list');

achievementsList.innerHTML = '';

gameConfig.achievements.forEach(achievement => {

const achievementItem = document.createElement('div');

achievementItem.className = `achievement-item ${achievement.completed ? 'completed' : ''}`;

achievementItem.innerHTML = `

<div class="achievement-icon">${achievement.icon}</div>

<div class="achievement-details">

<h4>${achievement.name}</h4>

<p>${achievement.description}</p>

<div class="achievement-status">

${achievement.completed ? '✅ Completo' : '❌ Incompleto'}

</div>

</div>

`;

achievementsList.appendChild(achievementItem);

});

}

// Renderiza as conquistas na tela de conquistas

function renderAchievements() {

const achievementsGrid = document.querySelector('.achievements-grid');

achievementsGrid.innerHTML = '';

gameConfig.achievements.forEach(achievement => {

const achievementCard = document.createElement('div');

achievementCard.className = `card ${achievement.completed ? 'completed' : ''}`;

achievementCard.innerHTML = `

<div class="card-name">${achievement.name}</div>

<div class="card-icon">${achievement.icon}</div>

<div class="card-ability">${achievement.description}</div>

<div class="card-power">Status: ${achievement.completed ? 'Completo' : 'Incompleto'}</div>

`;

achievementsGrid.appendChild(achievementCard);

});

}

// Volta para o menu principal

function backToMenu() {

showScreen('mainMenu');

resetGame();

}

// Reseta o jogo para um novo duelo

function resetGame() {

gameConfig.players[0].score = 0;

gameConfig.players[1].score = 0;

gameConfig.discardPile = [];

gameConfig.gameState = 'waiting';

gameConfig.currentPlayer = 0;

gameConfig.selectedCard = null;

document.getElementById('battleArea').innerHTML = '';

document.getElementById('gameLog').innerHTML = '';

setupDecks();

}

// Define a dificuldade do jogo

function setDifficulty(difficulty) {

gameConfig.difficulty = difficulty;

// Atualiza botões de dificuldade

document.getElementById('easyBtn').classList.remove('active');

document.getElementById('mediumBtn').classList.remove('active');

document.getElementById('hardBtn').classList.remove('active');

document.getElementById(`${difficulty}Btn`).classList.add('active');

updateGameLog(`Dificuldade definida para: ${difficulty === 'easy' ? 'Aprendiz' : difficulty === 'medium' ? 'Mago' : 'Arquimago'}`);

}

</script>

</body>

</html>