Medieval-Deck Technical & Mechanics Doc

Generated 2025-07-26

Este documento descreve a arquitetura de codigo e a implementacao das mecanicas de jogo para Medieval-Deck, sem cobrir release ou marketing.

1 Estrutura de pastas

```
medieval-deck/
 assets/ (arte gerada)
    bg/        backgrounds
    sheets/    sprite-sheets
    ui/        hud, icones
 scripts/
    gen_assets.py
 src/
    engine/    util generico
    game/      regras puras
    ui/        telas pygame
    main.py
 tests/
```

2 Modulo engine

Camada independente das regras; reutilizavel.

- animation.FrameAnimation frames, fps, loop, current(), done
- tween.Tween de/para valor, easing
- particles.ParticleEmitter gera dic {pos,vel,life}
- resources.load_texture(path) memoiza Surface

3 Pipeline de assets (SDXL)

scripts/gen_assets.py controla todo output IA.

- Le prompts.yaml -> gera PNG/ sprite-sheet
- Cache por hash(prompt+seed)
- Funcoes geradoras: generate_background, generate_sprite_sheet, generate_image

4 Modulos de jogo (src/game)

Todos **sem** dependencia pygame; testaveis.

- cards.py Card(name, cost, dmg=0, block=0, heal=0, status=[])
- deck.py Deck(draw_pile, hand, discard). draw(n), play(idx), shuffle()
- enemy.py Enemy(name, hp, intents=[(atk,8)...])
- combat.py CombatEngine(player, enemies) com fases: player_turn, enemy_turn, cleanup

5 Fluxo de combate (Engine + UI)

1. CombatScreen.update(): recebe dt
2. Input: click carta -> ui.hand.on_click(idx)
3. ui chama game.CombatEngine.play_card(idx)
4. Engine retorna efeito (dano, block)

5. ui.spawn_particles(target_pos) e altera anima player
6. EndTurn: Engine.enemy_turn(); intents resolvidas.
7. Loop ate victory/defeat flag.

6 Integracao de animacao

Cada sprite-sheet = 1 Surface horizontal. engine.animation corta frames.
Exemplo cavaleiro:

```
sheet = load_texture('assets/sheets/knight_idle.png')
idle = FrameAnimation(sheet, frames=10, fps=30, loop=True)
player.anim = idle
```

7 Progresso no mapa

game.map.Node(type, data) ligado em lista. MapScreen renderiza icones e seta atual.
Ao vencer combate, GameState.next_node() avanca ponteiro; se tipo 'reward', carrega RewardScreen.

8 Telas pygame (src/ui)

- ScreenBase handle_event, update(dt), draw()
- MenuScreen botao Start
- MapScreen icones de node, click avancar
- CombatScreen camada BG, mid (sprites), ui (mao/HUD)
- RewardScreen escolha 1 de 3 cartas

9 Sistema Save/Load

GameState dataclass com deck, hp, node_index, relics.
save_run() -> json.dump(GameState.__dict__)
load_run() -> instancia GameState e carrega tela correspondente.

10 Estrategia de testes

- pytest -q executa sem janela grafica usando SDL_VIDEODRIVER=dummy
- tests/test_deck.py exaustao, reshuffle
- tests/test_combat.py dano, block, intents
- tests/test_particles.py life decai