Desenvolvimento de Paciência (Solitaire)

Arquitetura, Algoritmos e Manual do Usuário

Pedro Franco de Camargo

Computação Gráfica

10 de dezembro de 2025

Roteiro da Apresentação

Introdução ao Projeto

O que é?

Uma implementação fiel do jogo *Klondike Solitaire* utilizando a linguagem Python e a biblioteca gráfica `pygame`. O projeto foca em arquitetura limpa e experiência de usuário fluida.

Objetivos Técnicos

- **Orientação a Objetos:** Uso de classes para Cartas, Jogo e Renderizador.
- **Loop de Jogo:** Separação clara entre *Input*, *Update* e *Draw*.
- **Matemática Vetorial:** Animações suaves (não lineares) para movimentação de cartas.
- **Gerenciamento de Estado:** Sistema robusto de desfazer (Undo) e controle de sementes (Seeds).

Estrutura de Arquivos (Modularização)

O projeto não é um script único, mas um sistema dividido em responsabilidades:

Núcleo Lógico:

- `game.py`: Contém a máquina de estados, pilhas de cartas e controle de fluxo.
- `card.py`: Objeto Carta com propriedades lógicas (valor, naipe) e físicas (x, y).
- `solitaire_rules.py`: Métodos estáticos que validam se um movimento é legal.

Interface e Recursos:

- `renderer.py`: Desenha o estado atual na tela (apenas leitura).
- `assets.py`: Recorta o *spritesheet* e carrega imagens.
- `main.py`: Gerencia o Menu e o Loop principal.

A Classe Carta: Lógica vs. Visual

Um dos grandes diferenciais é a separação das coordenadas. O Pygame usa inteiros (`int`) para desenhar, mas animações suaves exigem precisão decimal (`float`).

```
class Carta:
    def __init__(self, valor, naipe):
        # Estado Logico
        self.valor = valor
        self.naipe = naipe
        self.virada = False

        # Estado Fisico (Rect do Pygame - Inteiros)
        self.rect = pygame.Rect(0, 0, LARGURA, ALTURA)

        # Estado de Animacao (Float para suavidade)
        self.x_float = 0.0
        self.y_float = 0.0

```

Animação por Interpolação Linear (Lerp)

As cartas não se "teletransportam". Elas deslizam até o destino. Isso é feito recalculando a posição a cada frame.

Fórmula da Suavização

Para mover de P_{atual} para $P_{destino}$:

$$P_{novo} = P_{atual} + (P_{destino} - P_{atual}) \times \alpha$$

Onde α é o fator de velocidade (ex: 0.2).

Efeito: A carta começa rápida e desacelera suavemente ao chegar no alvo ("Ease-out effect"), criando uma sensação de polimento visual.

Sistema de Seeds (Sementes)

Para permitir que competições justas ou repetição de jogos, o embaralhamento não é totalmente aleatório, mas determinístico baseado em uma *Seed*.

```
# No arquivo game.py
if seed_usuario:
    self.seed = seed_usuario.upper()
else:
    # Gera uma string aleatoria (ex: "X7B9A")
    self.seed = ''.join(random.choices(chars, k=6))

# Fixa a aleatoriedade antes de embaralhar
random.seed(self.seed)
self.baralho = self._criar_baralho()
random.seed(None) # Libera para o resto do sistema

```

Sistema de Desfazer (Undo) - Tecla Z

O jogo utiliza uma **Pilha de Histórico**. Antes de qualquer movimento válido, o estado inteiro do jogo é clonado.

Desafio Técnico: Referências de Memória

Em Python, listas são passadas por referência. Simplesmente salvar `estado = self.mesa` não funciona, pois alterações futuras alterariam o histórico.

Solução: Uso de `copy.deepcopy()`.

```
1 def salvar_estado(self):
2     estado = {
3         'mesa': copy.deepcopy(self.pilhas_mesa),
4         'compra': copy.deepcopy(self.monte_compra),
5         # ... salva tudo
6     }
7     self.historico.append(estado)
8
```

Ao pressionar **Z**, o jogo restaura o último estado salvo e recalcula as posições visuais para evitar "pulos"gráficos.

Lógica do Clique Duplo (Automação)

Funcionalidade de qualidade de vida. Ao clicar duas vezes rapidamente em uma carta:

Fluxo de Verificação:

1. O sistema verifica se a carta é elegível para subir para as **Fundações** (Ases e sequências de naipes).
2. Se sim, move e anima a carta automaticamente.
3. Se não, verifica se ela pode ser movida para outra pilha da **Mesa**.
4. Isso acelera o final do jogo, evitando arrastar cartas óbvias manualmente.

Controles do Teclado e Mouse

Comando	Ação
Clique Esquerdo	Selecionar carta ou virar carta do monte.
Arrastar (Drag)	Mover cartas entre pilhas.
Clique Duplo	Tentar movimento automático inteligente.
Tecla 'Z'	**Desfazer:** Reverte a última jogada.
Tecla 'R'	**Reiniciar:** Volta ao menu principal (desiste).
Ctrl + V	Cola uma Seed no menu inicial.

Menu Principal e Configurações

O jogo inicia em um menu robusto que oferece:

- **Modo Fácil:** Vira 1 carta do monte de compra por vez. Ideal para iniciantes.
- **Modo Médio:** Vira 3 cartas por vez. É a regra clássica e mais difícil.
- **Input de Seed:** Caixa de texto onde o jogador pode digitar ou colar um código para jogar um cenário específico.
- **Tema de Cartas:** Botão para alterar a estampa do verso do baralho (carregado dinamicamente do `spritesheet`).

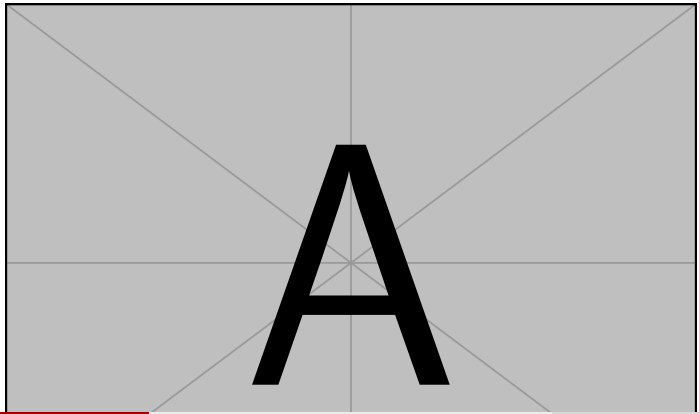
Interface durante o Jogo (HUD)

A tela de jogo (`renderer.py`) exibe informações cruciais:

- **Cronômetro:** Tempo decorrido desde o início da partida.
- **Seed Atual:** Mostrada no canto inferior direito.
- **Botão Copiar:** Ao lado da Seed, permite copiar o código para a área de transferência e enviar a um amigo.
- **Mensagens de Estado:** *Overlays* de "Vitória"ou "Game Over"(se implementado) com transparência alfa.

Tutorial: Objetivo do Jogo

Objetivo Final: Mover todas as 52 cartas para as 4 pilhas superiores direitas (chamadas de *Fundações*), organizadas por naipe (Copas, Espadas, Ouros, Paus) e em ordem ascendente (Ás $\rightarrow$ Rei).



Tutorial: Regras de Movimentação

Na Mesa (Pilhas Inferiores):

- Você deve construir sequências **Decrescentes**.
- Deve alternar as **Cores** (Vermelho sobre Preto, Preto sobre Vermelho).
- *Exemplo:* Um 6 Vermelho só pode ser colocado sobre um 7 Preto.

Movendo Grupos:

- Você pode arrastar uma pilha inteira de cartas, desde que elas já estejam organizadas corretamente entre si.

Espaços Vazios:

- Apenas o **Rei (K)** pode ser colocado em um espaço vazio na mesa.

Tutorial: Estratégias Básicas

1. **Priorize a Mesa:** Tente revelar as cartas viradas para baixo na mesa antes de comprar cartas do monte.
2. **Cuidado com as Fundações:** Não suba cartas para as fundações cedo demais se elas forem necessárias para segurar outras cartas na mesa.
3. **Gerencie os Reis:** Não esvazie uma coluna se você não tiver um Rei para colocar lá imediatamente.
4. **Use o Undo (Z):** O jogo permite experimentação. Se travou, volte e tente outro caminho.

Conclusão e Resultados

O projeto resultou em um jogo completo e polido.

- **Performance:** Roda a 144 FPS estáveis devido à otimização do `draw`.
- **Modularidade:** O código está pronto para expansão (novos modos, skins, sons).
- **Experiência:** A combinação de animações Lerp e controles responsivos oferece uma sensação profissional ("Game Juice").

Obrigado!

Dúvidas?