Project Overview

- **Game:** Scoundrel (Digital Edition) - Tactical Dungeon Crawler Card Game
- **Platform:** Mobile (Portrait Mode)
- **Engine:** Unity 6000.3.6f1
- **UI System:** Screen Space Canvas (uGUI)
- **Dependencies:** PrimeTween (animations), UniTask (async), ZLinq, New Input System, Init-Args (DI)

Technical Decisions

Decision	Choice	Rationale
Dependency Injection	Init-Args	Clean service registration, automatic resolution, better testability
UI Framework	Screen Space Canvas (uGUI)	Standard Unity UI, good mobile support, cards as UI Image components
Testing Strategy	Implementation first	Focus on getting mechanics working, add tests as polish

---

Game Design Summary

Deck Composition (44 Cards)

Type	Suit	Count	Values
Monsters	Spades	13	2-10, J(11), Q(12), K(13), A(14)
Monsters	Clubs	13	2-10, J(11), Q(12), K(13), A(14)
Shields	Diamonds	9	2-10 (no face cards)
Potions	Hearts	9	2-10 (no face cards)

Core Mechanics

1. **Static Shield** - Permanent armor (never degrades), only replaced by new diamonds
2. **Elemental Affinity Combat:**

    - Spades: 100% block → `Damage = Max(0, Monster - Shield)`
    - Clubs: 50% block → `Damage = Max(0, Monster - Floor(Shield/2))`
3. **Overdose System** - After using potion, other hearts lock until non-heart interaction
4. **Run Mechanic** - Costs 1 HP, moves cards to deck bottom, requires 3-4 cards, no consecutive runs

---

Design Patterns

Pattern	Usage	Justification
Command	Player actions	Encapsulates FightMonster, EquipShield, DrinkPotion, Run
State	Game flow	DealingState → PlayerTurnState → ProcessingState → GameOverState
Observer	Event bus	Decouples UI from game logic via C# events
Strategy	Damage calc	Different formulas for Spades (100%) vs Clubs (50%)
Factory	Card/Deck creation	Consistent 44-card deck generation

---

System Architecture

Core Services (with Init-Args DI)

```
None
[Service] GameManager      → State machine, game lifecycle
[Service] GameEvents       → Event bus (IGameEvents)
[Service] PlayerState      → HP, Shield, HeartLock, RunAvailable
[Service] DeckSystem       → Shuffle, draw, return to bottom
[Service] RoomSystem       → Current 4-card room management
[Service] CommandProcessor → Validate and execute commands
```

UI Components (uGUI)

```
None
Canvas (Screen Space - Overlay)
├── HUD
│   ├── HealthDisplay (♥ 15/20)
│   └── ShieldDisplay (♦ 8)
├── RoomPanel (2x2 Grid Layout)
│   └── CardSlot[0-3] → CardView (Image + Button)
├── Footer
│   ├── DeckCounter (32)
│   └── RunButton ("RUN -1 HP")
└── Dialogs (overlay)
```

Folder Structure

```
None
Assets/_Project/Scripts/
├── Core/
│   ├── Data/
│   │   └── CardData.cs
│   ├── Enums/
│   │   ├── CardSuit.cs
│   │   ├── CardRank.cs
│   │   ├── CardType.cs
│   │   ├── GameState.cs
│   │   └── GameResult.cs
│   ├── Interfaces/
│   │   ├── ICommand.cs
│   │   ├── IGameEvents.cs
│   │   ├── IPlayerState.cs
│   │   ├── IDeckSystem.cs
│   │   ├── IRoomSystem.cs
│   │   ├── IDamageCalculator.cs
│   │   └── IGameSettings.cs
│   ├── Services/
│   │   ├── GameManager.cs
```

```
|   |   ├── GameEvents.cs
|   |   ├── PlayerState.cs
|   |   ├── DeckSystem.cs
|   |   ├── RoomSystem.cs
|   |   └── CommandProcessor.cs
|   ├── Commands/
|   |   ├── FightMonsterCommand.cs
|   |   ├── EquipShieldCommand.cs
|   |   ├── DrinkPotionCommand.cs
|   |   └── RunCommand.cs
|   ├── Combat/
|   |   ├── SpadesDamageCalculator.cs
|   |   └── ClubsDamageCalculator.cs
|   └── Factory/
|       └── DeckFactory.cs
├── UI/
|   ├── Views/
|   |   ├── CardView.cs
|   |   ├── RoomView.cs
|   |   ├── HUDController.cs
|   |   ├── HealthDisplay.cs
|   |   ├── ShieldDisplay.cs
|   |   ├── RunButtonController.cs
|   |   └── DeckView.cs
|   ├── Dialogs/
|   |   ├── DialogService.cs
|   |   └── ConfirmDialog.cs
|   └── Animation/
|       ├── CardAnimator.cs
|       └── HUDAnimator.cs
├── Input/
|   └── CardInputHandler.cs
└── ScriptableObjects/
    ├── GameSettings.cs
    └── CardDatabase.cs
```

Implementation Phases

Phase 1: Foundation (Data & Interfaces)

Files to create:

- `Core/Enums/CardSuit.cs` - Spades, Clubs, Diamonds, Hearts
- `Core/Enums/CardRank.cs` - Two=2 through Ace=14
- `Core/Enums/CardType.cs` - Monster, Shield, Potion
- `Core/Enums/GameState.cs` - Initializing, Dealing, PlayerTurn, Processing, GameOver
- `Core/Enums/GameResult.cs` - None, Victory, Defeat
- `Core/Data/CardData.cs` - Readonly struct (Suit, Rank, computed Value & Type)
- `Core/Interfaces/*.cs` - All interface contracts
- `ScriptableObjects/GameSettings.cs` - Configuration values
- `ScriptableObjects/CardDatabase.cs` - Sprite mappings

Phase 2: Core Game Services

Files to create:

- `Core/Services/GameEvents.cs` - Event bus implementation
- `Core/Services/PlayerState.cs` - HP/Shield/Locks with event firing
- `Core/Factory/DeckFactory.cs` - Creates 44-card deck
- `Core/Services/DeckSystem.cs` - Shuffle, draw, return operations
- `Core/Services/RoomSystem.cs` - Room card management
- `Core/Combat/SpadesDamageCalculator.cs` - 100% efficiency formula
- `Core/Combat/ClubsDamageCalculator.cs` - 50% efficiency formula

Phase 3: Command System

Files to create:

- `Core/Commands/FightMonsterCommand.cs` - Fight logic + unlock hearts
- `Core/Commands/EquipShieldCommand.cs` - Set shield + unlock hearts
- `Core/Commands/DrinkPotionCommand.cs` - Heal + lock hearts
- `Core/Commands/RunCommand.cs` - Pay HP + move cards + disable run
- `Core/Services/CommandProcessor.cs` - Execute commands

Phase 4: Game Manager & State Machine

Files to create:

- `Core/Services/GameManager.cs` - Main orchestrator with state machine

State Flow:

```
None
Initialize → Deal → PlayerTurn ⟷ Processing → Deal (loop)
                        ↓
                   GameOver (Win/Loss)
```

Phase 5: Basic UI (uGUI)

Prefabs to create:

- `Prefabs/UI/CardPrefab.prefab` - Card Image + Button component
- `Prefabs/UI/HUD.prefab` - Health/Shield displays

Files to create:

- `UI/Views/CardView.cs` - Card visual + interaction
- `UI/Views/RoomView.cs` - 2x2 GridLayoutGroup manager
- `UI/Views/HUDController.cs` - Orchestrates HUD elements
- `UI/Views/HealthDisplay.cs` - HP text (♥ 15/20)
- `UI/Views/ShieldDisplay.cs` - Shield text (♦ 8)
- `UI/Views/RunButtonController.cs` - Button state management
- `UI/Views/DeckView.cs` - Remaining cards counter

Phase 6: Input & Interaction

Files to create:

- `Input/CardInputHandler.cs` - Tap/hold detection

Wiring:

- Card tap → Create appropriate command → Execute
- Run button → RunCommand → Execute

Phase 7: Animation & Feedback

Files to create:

- `UI/Animation/CardAnimator.cs` - Deal/flip/discard animations (PrimeTween)
- `UI/Animation/HUDAnimator.cs` - HP punch, shield flash

Visual feedback:

- Damage preview on monster hold (red -X)
- Shield efficiency indicator (Clubs dim the shield)
- Heart lock visual (greyed out + lock icon)

Phase 8: Dialogs & Game End

Files to create:

- `UI/Dialogs/DialogService.cs` - Dialog management
- `UI/Dialogs/ConfirmDialog.cs` - Shield downgrade prompt

Prefabs to create:

- `Prefabs/UI/ConfirmDialog.prefab`
- `Prefabs/UI/GameOverPanel.prefab`

Phase 9: Mobile Polish

- Safe area handling (notch/home indicator)
- Object pooling for CardView instances
- Touch input optimization
- Portrait mode lock
- Performance profiling

Key Interfaces (with Init-Args)

csharp

```CSharp
// Service registration example
[Service(typeof(IGameEvents))]
public class GameEvents : IGameEvents { ... }

[Service(typeof(IPlayerState))]
public class PlayerState : IPlayerState { ... }
```

Verification Approach

During Implementation

- Console logging for state transitions
- Debug UI for HP/Shield/Room state
- Play-test each phase before moving on

Post-Implementation Testing

- Unit tests for damage calculators
- Unit tests for PlayerState bounds
- Integration tests for game loop
- Edge case testing (0 HP, run at 1 HP, etc.)

Files to Implement (Ordered)

Start Here (Phase 1):

1. `Core/Enums/CardSuit.cs`
2. `Core/Enums/CardRank.cs`
3. `Core/Enums/CardType.cs`
4. `Core/Enums/GameState.cs`
5. `Core/Enums/GameResult.cs`
6. `Core/Data/CardData.cs`
7. `Core/Interfaces/IGameEvents.cs`
8. `Core/Interfaces/IPlayerState.cs`
9. `Core/Interfaces/IDeckSystem.cs`
10. `Core/Interfaces/IRoomSystem.cs`
11. `Core/Interfaces/IDamageCalculator.cs`
12. `Core/Interfaces/IGameSettings.cs`
13. `Core/Interfaces/ICommand.cs`
14. `ScriptableObjects/GameSettings.cs`
15. `ScriptableObjects/CardDatabase.cs`

This foundation enables all subsequent phases to build upon solid type-safe contracts.