개발팀이 \*\*[Project: 도리천]\*\*의 전투 시스템을 단계별로 구현할 수 있도록 정리한 \*\*개발 마일스톤(Development Milestone)\*\*입니다.

이 계획은 **'핵심 로직 검증(Core Logic)' $\rightarrow$ '플레이 가능한 프로토타입(Playable)' $\rightarrow$ 'UX 및 연출 강화(Polish)'** 순서로 진행됩니다.

# [개발 계획서] 전투 시스템 구현 로드맵

프로젝트: 도리천 (Doricheon)

목표: 인과 연산(Causal Anchor) 및 절대 물리 법칙 시스템 구현

기간: 총 5단계 (Phase 1 ~ 5)

## 📅 Phase 1. 물리 엔진 구축 (Universal Physics)

**목표:** 직군이나 미션 없이, **속성과 속성이 만났을 때 반응이 정상 출력되는지** 확인한다.

### 1-1. 데이터 구조 정의 (Data Structure)

* **Enum 정의:**
  + ElementType: Entropy, Stasis, Gravity, Resonance, Paradox, Axiom
  + ReactionType: Fusion, ThermalShock, Plasma, AbsoluteZero, Overload, BlackHole, None
* **Class 설계:**
  + BattleEntity: 속성 슬롯(Slot 1, 2) 관리, 상태이상 관리.
  + SkillData: 스킬의 속성, 데미지 계수, 타겟 범위 데이터.

### 1-2. 반응 테이블 로직 구현 (Core Logic)

* **Reaction Manager:**
  + CheckReaction(TargetElement, AttackElement) 함수 구현.
  + GDD 3.1항의 **[조합 테이블]** 하드코딩.
  + **Unit Test:** 🔴+🟣 입력 시 💥(Fusion) 리턴 확인. 🔵+🟡 입력 시 ⚡(Overload) 리턴 확인.

### 1-3. 특수 속성 예외 처리

* **Special Logic:**
  + Paradox 속성 입력 시, 타겟 속성 무관하게 강제 성공 플래그 리턴.
  + Axiom 속성 입력 시, 타겟 속성 무관하게 보호막 로직 트리거.

✅ Phase 1 완료 기준:

콘솔(Console) 창에서 두 속성을 입력했을 때, 정확한 현상(Reaction) 텍스트가 출력되어야 함.

## 📅 Phase 2. 전투 루프 & 미션 게이지 (Game Loop)

**목표:** 3단계(관측-연산-확정) 턴 구조를 만들고, **게이지가 차오르는 로직**을 완성한다.

### 2-1. 턴 매니저 (Turn Manager)

* **Phase State Machine:** Observe $\rightarrow$ Execute $\rightarrow$ Realize 상태 전환 구현.
* **Phase 1 (관측):**
  + 랜덤으로 CurrentMission (혼돈/정적) 생성 및 UI 표시.
  + MissionGauge 변수 초기화 (0.0f).

### 2-2. 게이지 산정 알고리즘 (Scoring)

* **Add\_Gauge 함수 구현:**
  + GDD 5항의 알고리즘 적용.
  + 반응 성공(50pt), 단일 속성 일치(15pt), 패러독스(100pt), 공리(30pt) 분기 처리.
* **Phase 3 (확정):**
  + 턴 종료 시 MissionGauge >= 100 체크.
  + 성공/실패에 따른 더미(Dummy) 로그 출력.

✅ Phase 2 완료 기준:

플레이어가 스킬을 선택해 공격하면, 결과에 따라 게이지가 차오르고 턴 종료 시 성공/실패가 판정되어야 함.

## 📅 Phase 3. 직군 및 캐릭터 구현 (Class System)

**목표:** 실제 캐릭터 스탯과 스킬을 적용하여 **전투의 다양성**을 확보한다.

### 3-1. 직군 베이스 구현 (Class Base)

* **직군별 스탯 템플릿 작성:**
  + Interceptor: HP/DEF 비중 높게 설정, 어그로(Aggro) 수치 관리 로직.
  + Executor: ATK 비중 높게 설정.
  + Stabilizer: HealPower 스탯 추가.
  + Pathfinder: AGI(속도) 스탯 및 턴 게이지 로직.

### 3-2. 스킬 로직 구체화

* **스킬 효과 구현:**
  + 데미지 공식 적용: (ATK \* SkillFactor) - EnemyDEF.
  + 현상(Reaction) 발동 시 추가 효과 적용 (예: 💥핵융합 시 광역 데미지 로직).
  + 🔴화상, 🔵빙결, 🟣둔화 등 상태이상(Debuff) 실제 적용.

✅ Phase 3 완료 기준:

인터셉터가 맞고, 엑시큐터가 때려서 콤보를 터트리는 '파티 플레이'가 가능해야 함.

## 📅 Phase 4. UI/UX 및 가이드 (Visualization)

**목표:** 기획서의 핵심인 \*\*'직관성'\*\*을 시각적으로 구현한다.

### 4-1. 신호등 시스템 (Smart Highlight)

* **UI Logic:**
  + UpdateSkillUI(): 현재 CurrentMission 타입과 캐릭터의 스킬 속성을 비교.
  + 일치하는 스킬 버튼에 Looping Glow 애니메이션 활성화.
  + ⚪공리 스킬 아이콘 색상 변경 (카멜레온 로직).

### 4-2. 타겟팅 프리뷰 (Prediction)

* **OnHover Event:**
  + 마우스를 적에게 올렸을 때, CheckReaction()을 미리 돌려서 결과값 예측.
  + 적 머리 위에 Preview Canvas 띄우기 (예상 현상 아이콘 + 예상 획득 게이지).

### 4-3. 미션 연출 (Juice)

* **게이지 연출:** 게이지가 찰 때 기계적인 사운드와 함께 차오르는 연출 (DoTween 등 활용).
* **확정/실패 연출:** 화면 전체 포스트 프로세싱(Post-processing) 효과 (성공 시 밝음 / 실패 시 글리치 노이즈).

✅ Phase 4 완료 기준:

UI만 보고도 "어떤 버튼을 눌러야 하는지" 알 수 있어야 하며, 타격 시 게이지가 차는 타격감이 느껴져야 함.

## 📅 Phase 5. 폴리싱 및 밸런스 (Polish)

**목표:** 게임의 완성도를 높이고 재미를 조율한다.

### 5-1. 아트/사운드 리소스 연결

* 6대 현상(핵융합, 절대영도 등) 전용 VFX 연결.
* 직군별 타격/피격 모션 및 SFX 연결.

### 5-2. 밸런스 튜닝 (Tuning)

* Paradox의 자해 데미지 비율 조정 (Risk vs Reward).
* Axiom의 보호막 수치 및 게이지 획득량 미세 조정.
* 직군별 기본 스탯 오차 범위 조정.

### 👨‍💻 개발자를 위한 한마디

* **핵심은 Phase 1입니다.** '절대 물리 법칙' 테이블이 꼬이면 뒤의 모든 로직이 무너집니다. 반응 테이블(Reaction Table) 유닛 테스트를 꼼꼼하게 진행해 주세요.
* **UI는 친절해야 합니다.** 이 게임은 시스템이 복잡해 보일 수 있으므로, Phase 4의 '미리보기'와 '하이라이트' 기능이 튜토리얼 역할을 대신해야 합니다.