# 낮/밤 루프 시스템

## ■ 시스템 이름

낮/밤 루프 시스템 (Day/Night Loop System)

## ■ 목적

낮과 밤을 자동 또는 수동으로 전환하여, 게임 플레이의 두 가지 핵심 사이클(경영/탐색)을 명확하게 구분한다.

## ■ 기본 개념

- 낮과 밤은 일정 시간 동안 유지된다.  
- 시간이 지나면 자동으로 낮에서 밤, 밤에서 낮으로 전환된다.  
- GameManager가 시간의 흐름, 상태 전환, 이벤트 트리거를 관리한다.  
- UI에 현재 시간과 남은 시간이 표시된다.

## ■ 데이터 구조

public enum TimeState { Day, Night }  
  
public class GameManager : MonoBehaviour {  
 public TimeState currentTime;  
 public float dayDuration = 300f; // 낮 지속 시간 (초)  
 public float nightDuration = 180f; // 밤 지속 시간 (초)  
 private float timeLeft;  
}

## ■ 동작 흐름

1. 게임 시작 시 낮 상태(TimeState.Day)로 초기화  
2. 낮 타이머 시작 → 시간이 감소  
3. 시간이 0이 되면 가게관리로 전환  
4. 플레이어가 버튼을 누르면 밤 시간 시작  
5. 밤 타이머 시작 → 시간이 감소  
6. 시간이 0이 되면 다시 낮으로 전환 (1~6을 루프)

## ■ 조건/트리거

- 낮 타이머 0 → 가게관리 전환  
- 가게관리 버튼 클릭 → 밤 전환  
- 밤 타이머 0 → 낮 전환

## ■ 예상 결과

- 낮 시작: 커피 주문 및 제작 시스템 활성화  
- 가게 관리 시작 : 재화 및 재료 관리 시스템 활성화  
- 밤 시작: 좀비, 파밍, 탐색 시스템 활성화  
- 전환 시 UI 변경, 사운드/조명 변경, 상태 저장 가능

## ■ UI 연동

- 타이머 UI: 남은 시간 표시  
- 현재 시간 상태 UI: 낮/밤 텍스트 또는 아이콘  
- 상태 전환 시 전환 애니메이션 또는 효과

## ■ 기타 고려 사항

- 일시 정지 시 타이머 정지 기능 필요  
- 테스트 및 디버깅용 수동 전환 키 입력 (예: T 키)  
- 전환 시 저장 가능 여부 고려

## ■ 프로그래밍 연동 예상 함수

void Update() {  
 timeLeft -= Time.deltaTime;  
 if (timeLeft <= 0) SwitchTime();  
}  
  
void SwitchTime() {  
 if (currentTime == TimeState.Day) {  
 currentTime = TimeState.Night;  
 timeLeft = nightDuration;  
 // 밤 시작 처리  
 } else {  
 currentTime = TimeState.Day;  
 timeLeft = dayDuration;  
 // 낮 시작 처리  
 }  
}