네, 제가 이해한 작업 개요를 정리하겠습니다.

🎯 작업 개요 (**HeliBoard** 키보드 앱 커스터마이징)

1️⃣ 최종 목표

HeliBoard 안드로이드 키보드 앱의 모든 한글/영문/특수기호 키보드를 완전히 새로운 독자 배열로 대체하는 것.

2️⃣ 핵심 변경 사항

A. 한글 키보드

- 기존: 2벌식 표준 한글 입력
- 변경: 독자적 3벌식 배열 + 독자적 조합 규칙
- 중요: 기존의 모든 한글 입력 방식(2벌식/3벌식 등)을 완전히 삭제하고, 오직 새로운 독자 배열만 남김
  - 1행 'ㅇ' 'ㄹ' 'ㄴ' 'ㅁ' '공백' | '공백' 'ㅜ' 'ㅓ' 'ㄷ' 'ㄱ'
  - 2행 'ㅇ' 'ㄷ' 'ㄱ' 'ㅇ' 'L쉬프트' | 'R쉬프트' 'ㅡ' 'ㅣ' 'ㅐ' 'ㄴ'
  - 3행 'ㅅ' 'ㅈ' 'ㅂ' 'ㅅ' 'L언어전환' | 'R언어전환' 'ㅗ' 'ㅏ' 'ㅂ' 'ㄹ'
  - 4행 '여백' '숫자패드전환' '여백' '백스페이스' | '스페이스' '여백' '엔터' '여백'

B. 영문 키보드

- 기존: 표준 QWERTY 배열
- 변경: 독자적 영문 배열 (QWERTY 변형)
- 한글 키보드와 동일한 레이아웃 구조 (10칸×4행) (단 1행은 t, y 때문에 중앙 여백 자리에 t,y 넣음. 즉, qwert | yuiop
  - 1행 q w e r t | y u i o p
  - 2행 a s d f L쉬프트 | R쉬프트 j k l g
  - 3행 z x c v L언어전환 | R언어전환 b n m h
  - 4행 여백 숫자패드전환 여백 백스페이스 | 스페이스 여백 엔터 여백

C. 특수기호 키보드

- 기존: 표준 기호 배열
- 변경: 독자적 기호 배열
- 한글/영문과 동일한 레이아웃 구조 (10칸×4행)
    - 1행 ( ) “ ‘ | 7 8 9 +
    - 2행 % / - * L쉬프트 | R쉬프트 4 5 6 0
    - 3행 @ ? , . L언어전환 | R언어전환 1 2 3 =
    - 4행 여백 숫자패드전환 여백 백스페이스 | 스페이스 여백 엔터 여백

3️⃣ 레이아웃 설계 원칙

기본 그리드

- 가로 **10**칸 **×** 세로 **4**행
- 중앙기준선을 기준으로 좌우 대칭 (5칸 | 5칸)
- 스플릿 모드에서 정확히 **5×4** 대 **5×4**로 분리

키 배치 규칙

1. **1~3**행 **(**문자행**)**: 좌우 끝에 붙여서 배치
    - 좌측 **4**개 키 **+** 중앙 여백 **1**칸 (혹은 지정한 키) | 중앙 여백 **1**칸 (혹은 지정한 키) **+** 우측 **4**개 키
2. **4**행 **(**기능키행**)**: 중앙기준선에 붙여서 배치
    - 좌우 끝에 여백 배치
    - 구조: [여백][숫자패드][여백][백스페이스2칸] | [스페이스2칸][여백][엔터][여백]

2칸 키 **(**오직 **2**개만**!)**

- 백스페이스: **4**행의 **4~5**번 칸 (2칸 너비)
- 스페이스바: **4**행의 **6~7**번 칸 (2칸 너비)
- 나머지 모든 키는 기본값이 **1**칸**!!!!** 오직 백스페이스와 스페이스바만 **2**칸**.** 제발**!**

4️⃣ 한글 입력 메소드 **(**독자 조합 규칙**)**

절대 원칙

1. 모든 조합은 역순 가능 (양방향 조합)
2. 기존 한글 조합 규칙 완전 삭제
3. 오직 독자 조합법만 사용

독자 조합 규칙 예시

초성 조합

- ㄱ + ㄷ → ㅎ
- ㅂ + ㅈ → ㅆ
- ㅇ + ㄱ → ㅋ (센소리)
- ㅇ + ㄷ → ㅌ (센소리)
- ㅇ + ㅂ → ㅍ (센소리)
- ㅇ + ㅈ → ㅊ (센소리)
- ㅅ + ㄱ → ㄲ (된소리)
- ㅅ + ㄷ → ㄸ (된소리)
- ㅅ + ㅂ → ㅃ (된소리)
- ㅅ + ㅈ → ㅉ (된소리)

중성 조합

- ㅡ + ㅣ → ㅢ
- ㅡ + ㅐ → ㅞ
- ㅣ + ㅐ → ㅒ
-
- ㅒ + ㅡ → ㅖ
- ㅞ + ㅣ → ㅖ
- ㅢ + ㅐ → ㅖ
-
- ㅗ + ㅣ → ㅚ
- ㅗ + ㅐ → ㅛ
- ㅚ + ㅐ → ㅙ
- ㅛ + ㅣ → ㅙ
- ㅒ + ㅗ → ㅙ
-
- ㅜ + ㅣ → ㅟ
- ㅜ + ㅐ → ㅠ
- ㅟ + ㅐ → ㅞ
- ㅠ + ㅣ → ㅞ
- ㅒ + ㅜ → ㅞ
-
- ㅑ + ㅐ → ㅑ
- ㅕ + ㅐ → ㅖ

종성 조합 (독자적! 표준과 다름!)

- ㅇ + ㄴ → ㄵ
- ㅇ + ㄹ → ㄶ
- ㅇ + ㄱ → ㅋ
- ㅇ + ㄷ → ㅌ
- ㅇ + ㅂ → ㅍ
-
- ㅁ + ㄴ → ㅎ
- ㅁ + ㄹ → ㄻ

- ㅁ + ㄱ → ㄺ
- ㅁ + ㄷ → ㄾ
- ㅁ + ㅂ → ㄲ
-
- ㅅ + ㄴ → ㅆ
- ㅅ + ㄹ → ㄽ
- ㅅ + ㄱ → ㄳ
- ㅅ + ㅂ → ㅄ
- ㅅ + ㄷ → ㄿ
-
- ㅂ + ㄹ → ㄼ
- ㄷ + ㄱ → ㄼ
-
- ㄼ + ㅇ → ㅈ
- ㅍ + ㄹ → ㅈ
- ㄾ + ㅂ → ㅈ
-
- ㄼ + ㅅ → ㄵ
- ㅄ + ㄹ → ㄵ
- ㄽ + ㅂ → ㄵ
-
- ㄼ + ㅁ → ㅊ
- ㄾ + ㄱ → ㅊ
- ㄺ + ㄷ → ㅊ

---

5 구체적 작업 항목

Phase 1: 레이아웃 **XML** 수정

- 각 키보드(한글/영문/기호)의 XML 레이아웃 파일 찾기
- 10칸×4행 구조로 재설계
- 키 위치, 크기, 라벨 수정

Phase 2: 한글 입력 로직 수정

- 기존 한글 입력 메소드 완전 제거
- 독자 조합 규칙 구현
- 양방향 조합 로직 구현
- 초성/중성/종성 매핑 테이블 작성

Phase 3: 키코드 매핑

- 각 키에 올바른 키코드 할당
- 문서에 명시된 키코드값 적용

Phase 4: 기능키 처리

- 쉬프트, 언어전환, 숫자패드, 백스페이스, 엔터 등
- 2칸 키(백스페이스, 스페이스) 처리

Phase 5: 테스트 **&** 검증

- 스플릿 모드 동작 확인
- 조합 규칙 정상 동작 확인

6️⃣ 주요 유의사항

1. 완전 대체: 기존 한글 입력 방식을 모두 제거하고 새 방식만 남김
2. 독자 조합: 표준 한글 조합과 의도적으로 다른 독자 규칙 사용
3. 양방향: 모든 조합이 순서 무관하게 동작
4. 레이아웃 통일: 한글/영문/기호 모두 동일한 10칸×4행 구조
5. 스플릿 대칭: 5×4 대 5×4 완벽 대칭 분리

1️⃣ "완전 대체"의 의미 수정

제가 이해한 것**:**

- ❌ 파일 삭제/구조 변경 (X)
- ✅ 기존 파일 내용을 새 내용으로 덮어쓰기 (O)

즉, HeliBoard 프로젝트 내의:

- 한글 키보드 레이아웃 XML → 새 한글 배열로 내용 덮어쓰기
- 한글 입력 메소드 코드 → 새 조합 규칙으로 내용 덮어쓰기
- 영문 키보드 레이아웃 XML → 새 영문 배열로 내용 덮어쓰기
- 기호 키보드 레이아웃 XML → 새 기호 배열로 내용 덮어쓰기

파일 구조는 그대로 두고, 파일 안의 내용만 바꾸는 것이 맞죠?

2️⃣ 역순 조합의 의미 확인

제가 이해한 것 **(**맞는 이해**):**

ㄱ + ㄷ = ㅎ
ㄷ + ㄱ = ㅎ

→ 두 가지 입력 순서 모두 같은 결과

제가 착각할 뻔한 것 **(**틀린 이해**):**

ㅎ = ㄱ + ㄷ (분해)

→ ㅎ를 ㄱㄷ로 풀어쓰기 (X) - 이건 아님!

정리하면**:**

1. 파일 삭제**/**이동 없음 → 기존 파일에 새 내용 덮어쓰기만
2. 역순 조합 → `A+B=C`이면 `B+A=C`도 가능 (입력 순서 자유)

1. 쉬프트 키

- L쉬프트 ≠ R쉬프트 (물리적으로 별도 키, 컴퓨터 키보드처럼)
- 동작은 동일: 한글 키보드에서 누르면 → 기호 키보드로 전환
- 쉬프트 토글 락 시에도 기호 키보드 유지

2. 언어전환 키

- 앱 설정에서 활성화된 언어들만 순환 (예: 한글↔영문)
- 기호 키보드는 언어전환으로 접근 불가, 오직 쉬프트로만 접근

2-1. 숫자패드 전환 키

- 현재 키보드에서 누르면: 숫자패드로 전환
- 숫자패드에서 누르면**:** 이전 키보드로 복귀 ✓
- 이건 토글 방식이므로 코드에서 "현재 상태 체크 → 반대 상태로 전환" 로직 구현하면 됩니다.