〈Rewired〉

1. 한 줄 요약 / 핵심 장점

Rewired는 Unity의 기본 Input Manager 또는 새 Input System보다 훨씬 강력한 입력 처리 패키지입니다. 플랫폼/컨트롤러 호환성, 커스터마이징, 멀티 컨트롤러 지원, 런타임 리매핑 등 실무 수준의 입력 시스템 구축에 적합합니다.

2. 적용 대상

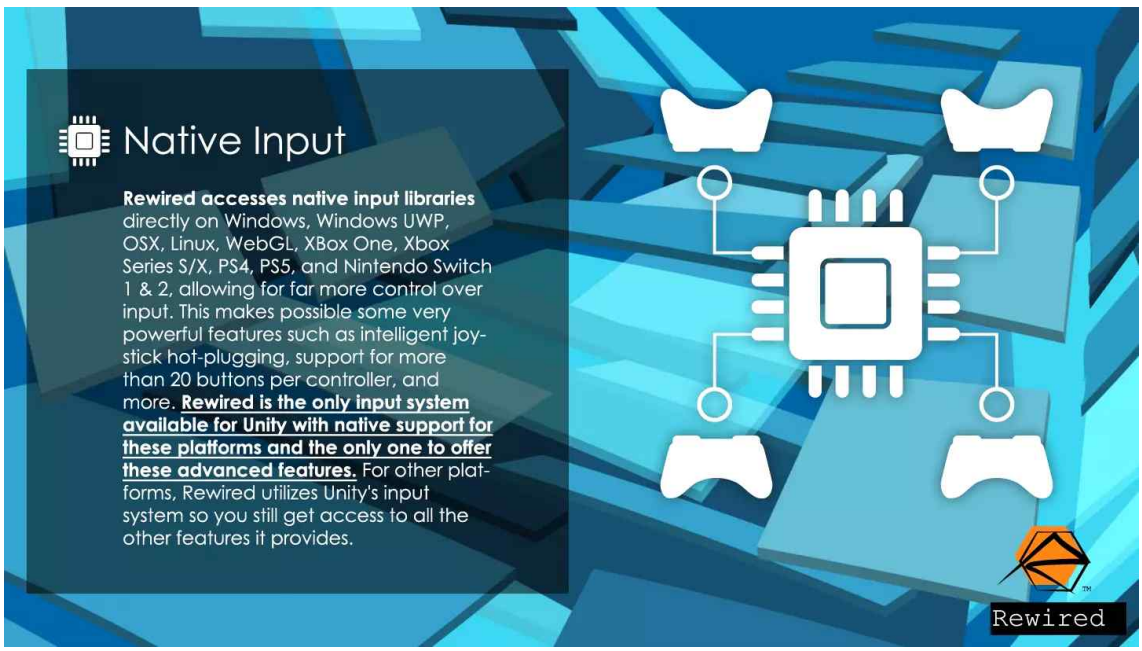
– 콘솔 / PC / 모바일 멀티 플랫폼 지원 필요할 때

– 여러 종류의 컨트롤러(XInput, DualShock, Switch Pro, 아케이드 스틱 등) 대응해야 할 때

– 플레이어별로 멀티 입력(로컬 멀티플레이) 관리가 필요할 때

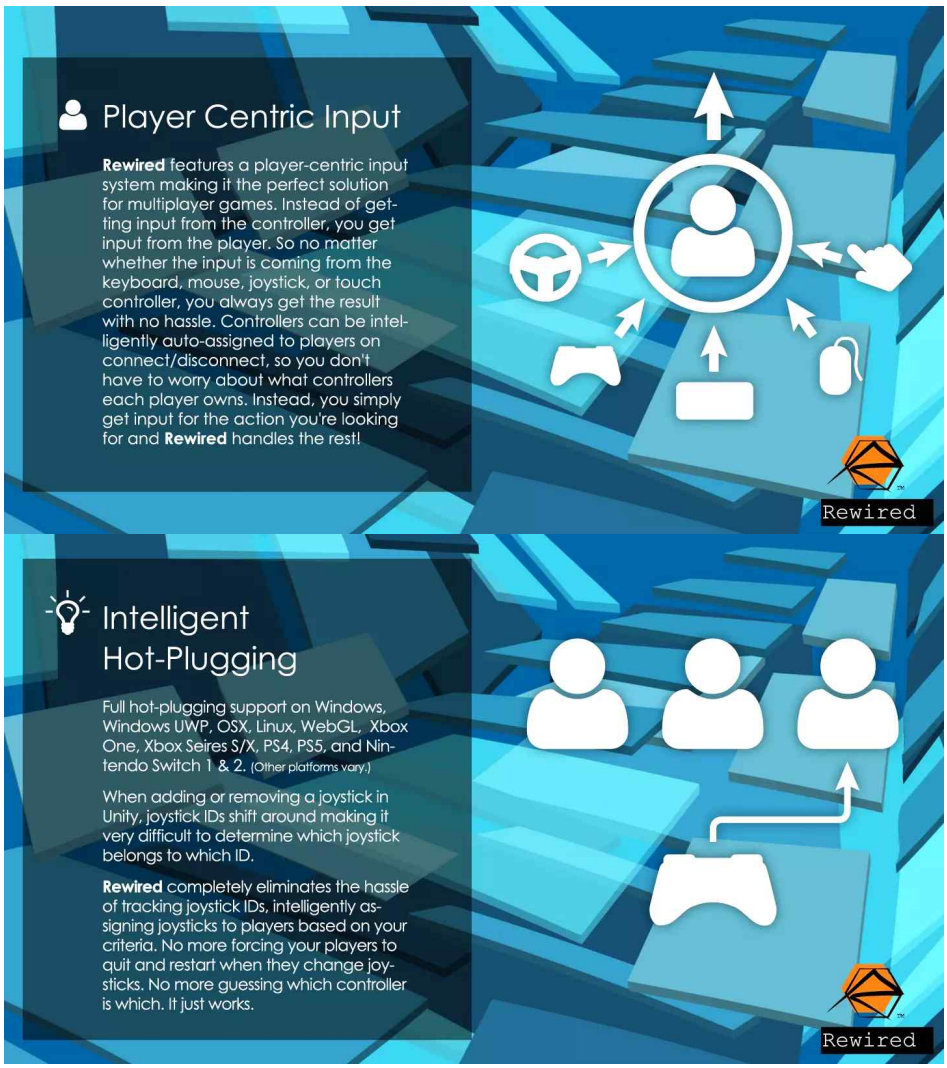
– 런타임 키 리매핑, 진동(루머), 압력 감지 등 고급 입력이 필요할 때

– Unity 기본 Input Manager의 한계(고정된 매핑, 이벤트 처리 부족 등)를 벗어나고 싶을 때



3. 주요 기능

기능	설명
컨트롤러 자동 인식	400개 이상의 컨트롤러 자동 지원 (XInput, DirectInput, PS4/PS5, Switch Pro 등)
플레이어 시스템	여러 Player 객체를 관리〉 로컬 멀티플레이 지원
액션 기반 입력(Action System)	입력을 “Jump”, “Shoot” 등 추상화된 액션으로 관리 (버튼 이름이 아니라 의미 단위로 접근)
런타임 리매핑	게임 내에서 키/버튼/축 매핑 변경 가능
고급 기능	Force Feedback(진동), 압력 감지, Unity UI Navigation 통합
크로스 플랫폼	Windows, Mac, Linux, iOS, Android, 콘솔 등 다양한 환경 지원



4. 기본 설치 및 세팅

4.1 Unity Asset Store〉 Rewired (유료) 구매 후 Import

4.2 'Rewired Input Manager' Prefab 씬에 추가

- (Project 〉 Rewired 〉 Prefabs 폴더 내 존재)

4.3 'Rewired Input Manager'〉 Inspector에서 Players, Maps, Actions 설정

- Players: 입력을 받을 캐릭터 단위
- Actions: Jump, Move, Fire 같은 의미 단위 입력
- Maps: 특정 컨트롤러에 Action을 매핑한 집합

5. 코드 예제

5.1 플레이어 가져오기

```
using UnityEngine;
using Rewired;

public class PlayerController : MonoBehaviour
{
    private Player player; // Rewired Player

    void Start()
    {
        // ID 0번 플레이어 가져오기
        player = ReInput.players.GetPlayer(0);
    }

    void Update()
    {
        if (player.GetButtonDown("Jump"))
        {
            Debug.Log("Jump!");
        }

        float move = player.GetAxis("MoveHorizontal");
        transform.Translate(Vector3.right * move * Time.deltaTime * 5f);
    }
}
```

5.2 여러 플레이어 지원 (로컬 멀티플레이)

```
void Start()
{
    for (int i = 0; i < ReInput.players.playerCount; i++)
    {
        Debug.Log("Player " + i + " is ready");
    }
}
```

5.3 런타임 키 리매핑

```
// Jump 액션을 Space 키로 매핑 변경
player.controllers.maps.SetMapsEnabled(true, "Default");
player.controllers.maps.GetMap(ControllerType.Keyboard, 0).ReplaceElementMap(
    player.controllers.maps.GetMap(ControllerType.Keyboard,
0).GetFirstElementMapWithAction("Jump"),
    new KeyCodeSourceMap(KeyCode.Space));
```

6. Unity 기본 입력 시스템 vs Rewired

항목	Unity Input Manager	Unity New Input System	Rewired
멀티 컨트롤러	제한적	개선됨	완벽 지원
런타임 리매핑	불가	기능	고급
플랫폼 호환성	제한적	개선됨	매우 강력
로컬 멀티플레이	불가	기능	플레이어 단위 시스템 내장
UI 통합	제한적	있음	확장 가능
안정성/최적화	기본 수준	개선됨	실무 검증됨

7. 실무 팁 & Best Practices

- Action 기반 설계: 입력을 “버튼 이름(KeyCode)” 대신 “Jump, Fire, Run” 같은 Action 단위로 관리 → 플랫폼/컨트롤러 변경에도 코드 수정 최소화.
- 플레이어 단위 관리: 로컬 멀티플레이 시 Player ID를 확실히 나누고, 각자 컨트롤러 연결 관리.
- UI Navigation 연계: Unity UI 이벤트 시스템과 연결하면 메뉴 네비게이션도 동일 시스템으로 처리 가능.
- 리소스 정리: Player 수가 많거나 다양한 맵을 쓸 경우, ScriptableObject 또는 CSV로 매핑 관리하면 유지보수 쉬움.
- 퍼포먼스 주의: 매 프레임 ‘ReInput’ 호출은 빠르지만, 대량 루프 시 캐싱 권장.

9) 요약 & 추천 사용 시나리오

- AAA 콘솔/PC 멀티플랫폼 게임: 다양한 컨트롤러 대응
- 로컬 멀티플레이 게임: 플레이어별 입력 관리
- e스포츠/대회용 게임: 안정적인 컨트롤러 지원 필요
- 입력 커스터마이즈가 중요한 RPG/액션 게임

Rewired는 Unity 입력 시스템의 상위호환으로, 입력 문제가 프로젝트의 병목이 될 경우 필수적인 투자 가치가 있습니다.