# UE5의 Enhanced Input 시스템 - 키 입력 처리 과정 (타임라인 방식)

언리얼 엔진 5에서 **키보드 입력이 발생한 순간부터 `UInputComponent`로 전달되기까지의 내부 처리 과정**을 타임라인 방식으로 설명하겠습니다.

---

## 🔹 타임라인: 입력 발생부터 UInputComponent까지의 전달 흐름

아래는 **"플레이어가 키보드의 W 키를 눌러 캐릭터를 앞으로 이동시키는 과정"**을 기반으로 UE5의 내부 처리 과정을 설명한 것입니다.

## ⏱️ [T0] - 입력 장치에서 키 입력 발생

- 사용자가 키보드의 `W` 키를 누름.
- 운영체제(OS)가 해당 입력 이벤트를 감지함.
- OS는 입력을 `DirectInput`, `XInput`, `RawInput` 등의 API를 통해 응용 프로그램(언리얼 엔진)에 전달함.

## ⏱️ [T1] - 언리얼 엔진의 `FWindowsApplication` 입력 시스템이 이벤트 수신

- 언리얼 엔진은 `FWindowsApplication::ProcessMessage()`를 통해 OS로부터 키 입력 이벤트를 받음.
- 이 이벤트는 `FGenericApplicationMessageHandler` 인터페이스를 통해 언리얼 엔진 내부의 입력 시스템으로 전달됨.

cpp 복사 편집

```cpp
bool FWindowsApplication::ProcessMessage(UINT Msg, WPARAM wParam, LPARAM lParam)
{
    switch (Msg)
    {
        case WM_KEYDOWN:
        case WM_SYSKEYDOWN:
            MessageHandler->OnKeyDown(KeyCode);
            break;
    }
}
```

- `WM_KEYDOWN` 이벤트를 감지하고 `MessageHandler->OnKeyDown(KeyCode);`를 호출하여 언리얼의 입력 시스템으로 전달.

## ⏱️ [T2] - `FInputProcessorStack`에 입력 이벤트 전달

- 언리얼 엔진 내부에서 **입력 프로세서 스택**(FInputProcessorStack) 을 사용하여 입력을 관리.
- 스택 구조이므로 **현재 활성화된 입력 처리기**(Widgets, UI, PlayerController 등) 가 우선적으로 이벤트를 확인.

```cpp
FSlateApplication::Get().ProcessKeyDownEvent(InputKey);
```

- 이 단계에서, **UI가 먼저 키 입력을 받을지 결정** → UI가 입력을 사용하지 않으면, 게임 플레이어 입력으로 전달됨.

## ⏱️ [T3] - `UPlayerInput` 을 통해 입력을 `Enhanced Input` 시스템으로 전달

- UI에서 입력을 소모하지 않았다면, `UPlayerInput` 을 통해 입력이 전달됨.
- `UPlayerInput::InputKey()` 에서 키 입력을 처리하고, 바인딩된 `UEnhancedPlayerInput` 객체로 이벤트 전달.

cpp 복사 편집

```cpp
bool UPlayerInput::InputKey(FKey Key, EInputEvent Event)
{
    return EnhancedInputSubsystem->HandleKeyInput(Key, Event);
}
```

- `EnhancedInputSubsystem` 에서 현재 활성화된 **Input Mapping Context (IMC)** 를 기반으로 입력을 매핑함.

## ⏱️ [T4] - Enhanced Input Subsystem에서 입력 이벤트 매핑 및 변환

- `UEnhancedPlayerInput::ProcessInputStack()`이 호출되며, 현재 활성화된 입력 컨텍스트에서 입력을 확인함.
- `Input Mapping Context(IMC)`를 기반으로, 키 입력이 어떤 `Input Action`에 바인딩되었는지 확인.

cpp 복사 편집

```cpp
void UEnhancedPlayerInput::ProcessInputStack()
{
    for (const TPair<FInputActionInstance, FKey> Pair : ActiveInputActions)
    {
        if (Pair.Value == PressedKey)
        {
            Pair.Key->TriggerEvent(ETriggerEvent::Triggered);
        }
    }
}
```

- `Input Mapping Context`에서 키 입력이 특정 `Input Action` (`IA_MoveForward`)에 연결되어 있는지 확인.
- `ETriggerEvent::Triggered` 이벤트를 발생시켜 바인딩된 기능을 실행.

## ⏱️ [T5] - `UInputComponent`의 바인딩된 이벤트 실행

- `UInputComponent`에서 해당 입력이 바인딩된 함수를 호출함.
- `ACharacter::SetupPlayerInputComponent()`에서 `BindAction()`을 사용하여 바인딩된 함수 실행.

cpp 복사 편집

```cpp
void AMyCharacter::SetupPlayerInputComponent(UInputComponent* PlayerInputComponent)
{
    Super::SetupPlayerInputComponent(PlayerInputComponent);

    // 이동 입력 바인딩
    UEnhancedInputComponent* EnhancedInput = Cast<UEnhancedInputComponent>(PlayerInputComponer
    if (EnhancedInput)
    {
        EnhancedInput->BindAction(MoveForwardAction, ETriggerEvent::Triggered, this, &AMyChara
    }
}

void AMyCharacter::MoveForward(const FInputActionValue& Value)
{
    FVector Direction = GetActorForwardVector();
    AddMovementInput(Direction, Value.Get<float>());
}
```

- `MoveForwardAction`에 대한 입력이 발생하면 `MoveForward()` 함수가 실행됨.
- `AddMovementInput()`을 호출하여 캐릭터 이동을 처리.

## ⏱️ [T6] - 캐릭터 이동이 최종 반영됨

- `ACharacter::AddMovementInput()`이 호출되면, `CharacterMovementComponent`가 업데이트됨.
- `UCharacterMovementComponent::TickComponent()`에서 물리적으로 이동이 적용됨.

cpp 복사 편집

```cpp
void UCharacterMovementComponent::TickComponent(float DeltaTime, ELevelTick TickType, FActorC
{
    Super::TickComponent(DeltaTime, TickType, ThisTickFunction);

    // 최종 이동 벡터 적용
    ApplyMovement(DeltaTime);
}
```

- 물리 엔진을 통해 최종 위치가 갱신되며, `W` 키 입력이 캐릭터 이동으로 반영됨.

## 🔹 전체 타임라인 요약

| 단계 | 설명 |
| --- | --- |
| T0 | 사용자가 키보드의 `W` 키를 누름. |
| T1 | `FWindowsApplication`이 OS에서 키 입력을 수신. |
| T2 | `FInputProcessorStack`에서 입력을 UI 또는 게임 시스템으로 전달. |
| T3 | `UPlayerInput`을 통해 `Enhanced Input`으로 전달. |
| T4 | `Enhanced Input Subsystem`에서 `Input Mapping Context`를 기반으로 입력을 매핑. |
| T5 | `UInputComponent`에서 바인딩된 함수(`MoveForward()`) 실행. |
| T6 | `UCharacterMovementComponent`에서 캐릭터 이동 반영. |

## 📌 결론

언리얼 엔진 5의 Enhanced Input 시스템은 **입력 이벤트를 보다 세분화하고, 유연한 입력 매핑을 제공**하여,

1️⃣ **멀티플랫폼(키보드, 컨트롤러, VR 등) 지원이 강화**

2️⃣ **입력 컨텍스트(Input Context)를 활용해 상태별 입력 관리가 가능**

3️⃣ **런타임에서 동적으로 입력을 변경할 수 있음**

이러한 과정을 통해 **보다 직관적이고 효율적인 입력 시스템을 구축**할 수 있습니다. 🚀