분석 문제: 분석한 내용을 직접 작성하고, 강의의 코드를 다시 한번 작성하며 복습해봅시다.

1. 입문 주차와 비교해서 입력 받는 방식의 차이와 공통점을 비교해보세요.

내용 정리

* 입문 주차: SendMassage를 이용한 입력 처리.
* 숙련 주차: Invoke Unity Events를 이용한 입력 처리.

공통점

* 스크립트에서 OnIput과 같은 함수를 이용해서 입력을 처리한다.
* 입력 방식에 대한 반응을 개발자가 원하는 대로 구성할 수 있다.

SendMassage에 비해서 **Invoke Unity Events가 가지는 이점**

* 다른 코드 없이 입력 시 여러 함수를 동시에 진행하도록 만들 수 있다.
* 스크립트가 PlayerInput 컴포넌트와 다른 객체에 있어도 사용 가능하다.
* 적용 타이밍을 정확하게 지정할 수 있다.
  + Started: 버튼을 누를 때.
  + Performed: 버튼 지속
  + Canceled: 버튼을 땔 때.

Invoke Unity Events에 비해서 **SendMassage가 가지는 이점**

* OnInput(InputValue value)로 선언되기만 하면 되서 코드가 간결하다.
* 함수를 인스펙터를 통해서 이벤트로 지정할 필요가 없다.

1. CharacterManager와 Player의 역할에 대해서 고민해보세요.

* CharacterManager는 Player에 대한 정보를 다른 클래스에서 사용할 수 있도록 하는 연결체 역할으로, 플레이어에 대한 정보를 싱글톤으로 들고 있음으로써 다른 클래스에서 Player 객체 정보를 가지고 있을 필요 없이 CharacterManager.Instance.Player 라는 코드 하나로 Player에 대해서 접근할 수 있도록 하는 역할입니다.
* Player는 CharacterManager와 유사하게 다른 객체에서 Player에 대한 정보를 확인하기 쉽도록 모아놓는 역할이지만, CharacterManager와는 달리 플레이어 객체에 직접 포함되어 플레이어의 상황에 따라서 달라지는 요소를 모두 적용할 수 있는 클래스입니다.
* 정리하면, CharacterManager는 Player라는 정보에 접근할 수 있도록 만들어주는 연결 매체라면, Player는 실질적인 요소(객체)를 담고 있는 DataContainer라고 할 수 있습니다.

1. 핵심 로직을 분석해보세요(Move, CameraLook, IsGrounded)

* Move
* 입력값이 들어오면 그것을 CurMovementInput에 반영한 뒤, FixedUpdate에서CurMovementInput 변수를 이용해서 이동을 구현합니다.
* 이동 방식은 Rigidbody.velocity를 입력값에 맞춰서 고정하는 방식으로, 이를 통해서 일정한 속도를 유지할 수 있도록 구성되어 있습니다.  
  다만, 이 경우 넉백이나 물체 상호작용에 의해서 발생하는 큰 충격을 처리하기 위해서는 AddForce 이외에도 별도의 다른 알고리즘을 같이 사용해야 한다는 난점이 발생하기도 하는 방식입니다.
* 만약 상승 혹은 하강 속도가 있다면 dir.y = rigidbody.velocity.y 라인을 통해서 그 속도를 유지하도록 만든 구성입니다.
* CameraLook
  + Move와 마찬가지로, 입력값이 들어오면 그것을 mouseDelta에 반영한 뒤, Update에서 mouseDelta 값을 이용해서 카메라 회전을 구현합니다.
  + X축 각도(가로를 기점으로 하므로 위아래로 회전한다.)는 최대 및 최소 한계가 존재하여 ‘정면’은 계속해서 유지되도록 구성이 되어 있고, Y축 각도(세로를 기점으로 하므로 좌우로 화전한다.)는 자유롭게 움직일 수 있는 구조입니다.
* IsGrounded
* 일정 주기로 바닥의 0.1 위에서 바닥을 향해서 길이 0.1의 Raycast를 작동, 이 결과에 따라서 지면에 서 있는지, 공중에 떠 있는지를 판단합니다.
* 만약에 Raycast 지점 사이에 첨단처럼 받치는 지형이 있다면 그에 대한 처리가 난감해진다는 단점이 있어, 캐릭터의 크기에 따라서 Raycast 정점의 배치와 양을 어떻게 구성할 지 조정해야할 수도 있습니다.
* 여담으로, 같은 방식을 정면을 향해서 작동하면 ‘벽’을 감지할 수 있습니다. 이에 대한 조작을 추가하여 등반을 구현할 수도 있습니다.

1. Move와 CameraLook 함수를 각각 FixedUpate, LateUpdate에서 호출하는 이유에 대해 생각해보세요.

Move는 캐릭터 객체가 이동하는 것이기 때문에 고정된 시간이 아니라면 넘실대듯이 움직이는 문제점이 발생합니다. 특히 Rigidbody의 이동 판정은 FixedUpdate와 동일한 타이밍에 발생하기 때문에 이에 대한 타이밍 문제 상으로도 FixedUpdate에서 처리하는 것이 적합합니다.

CameraLook은 화면을 돌리는 것으로, 화면 갱신은 Update와 동일한 타이밍에 발생하기 때문에 플레이어 조작과 맞춘 적절한 회전 각도를 나타내기 위해서는 Update에서 처리하는 것이 더욱 적합합니다.