유니티 프로그래밍 강의

이준

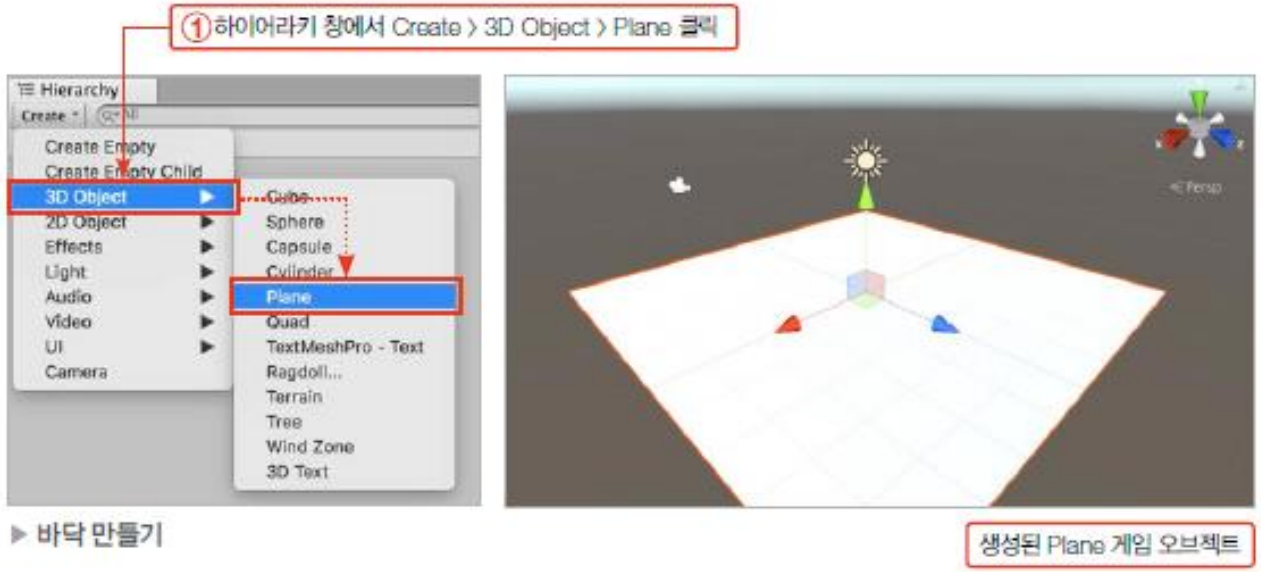
3D 닷지 플레이어 제작

닷지 플레이어 제작

- 씬 구성하기
  - 레벨의 바닥과 벽을 만듬

[과정 01] 바닥 만들기

① 바닥 Plane 게임 오브젝트 만들기(**하이어라키** 창에서 Create > 3D Object > Plane 클릭)

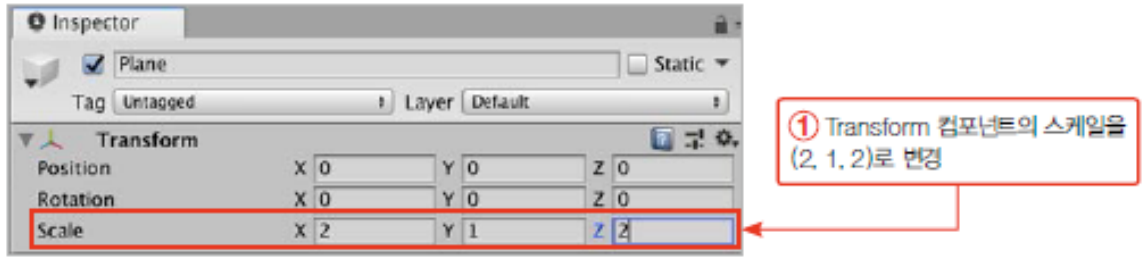


▶ 바닥 만들기

생성된 Plane 게임 오브젝트

[과정 02] 가로와 세로 길이를 두 배 늘리기

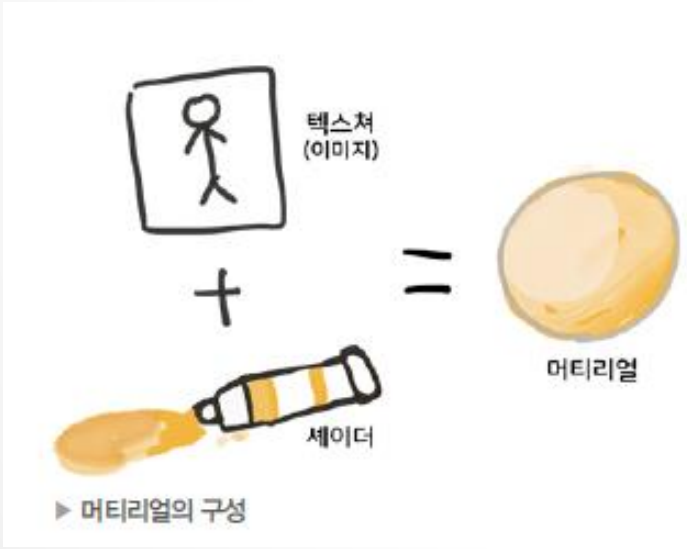
① **인스펙터** 창에서 Plane 게임 오브젝트의 **Transform** 컴포넌트의 **스케일**을 (2, 1, 2)로 변경



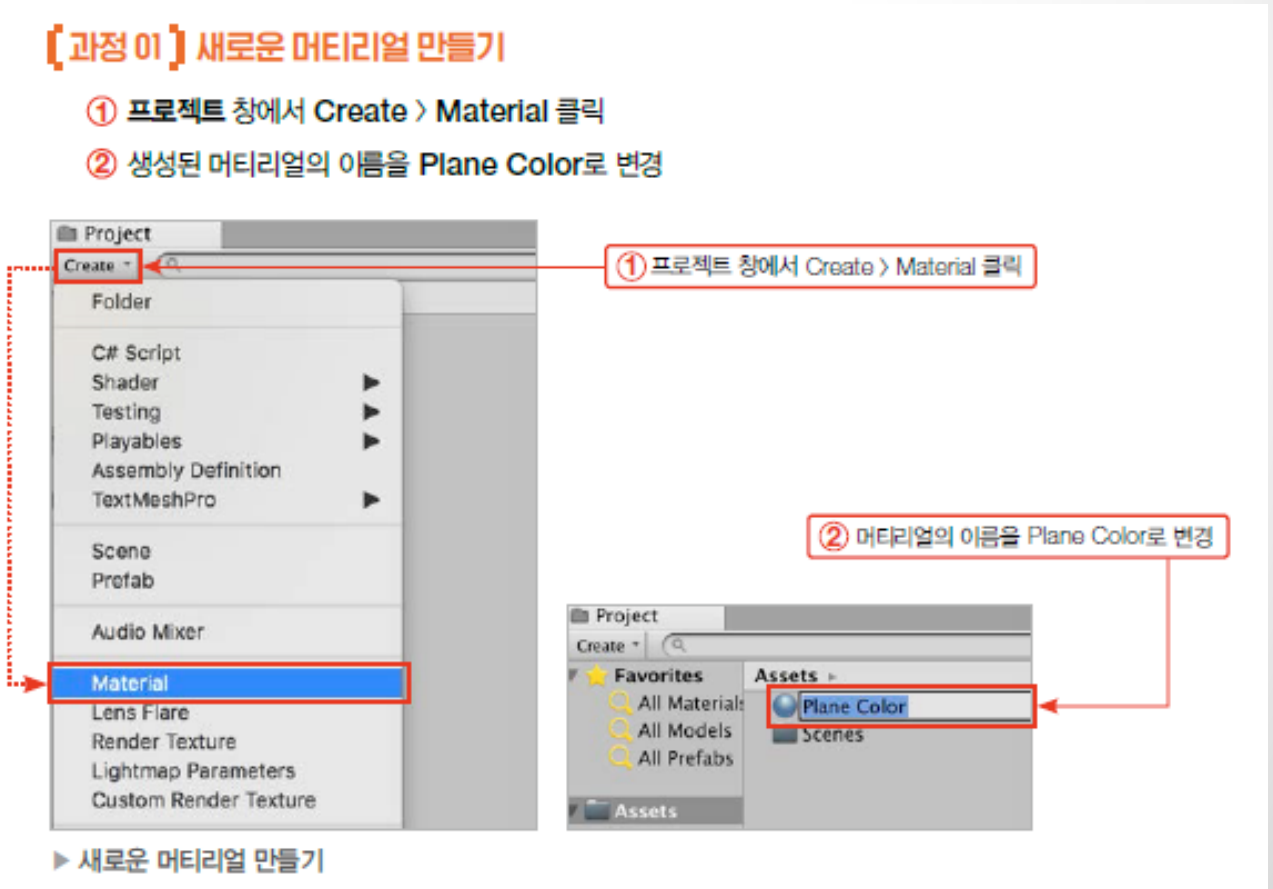
▶ 가로와 세로 길이를 두 배 늘리기

닷지 플레이어 제작

- 유니티에서 게임 오브젝트의 컬러는 **머티리얼**이 결정함.
- 머티리얼은 셰이더와 텍스처가 합쳐진 에셋으로, 오브젝트의 픽셀 컬러를 결정함.



▶ 머티리얼의 구성



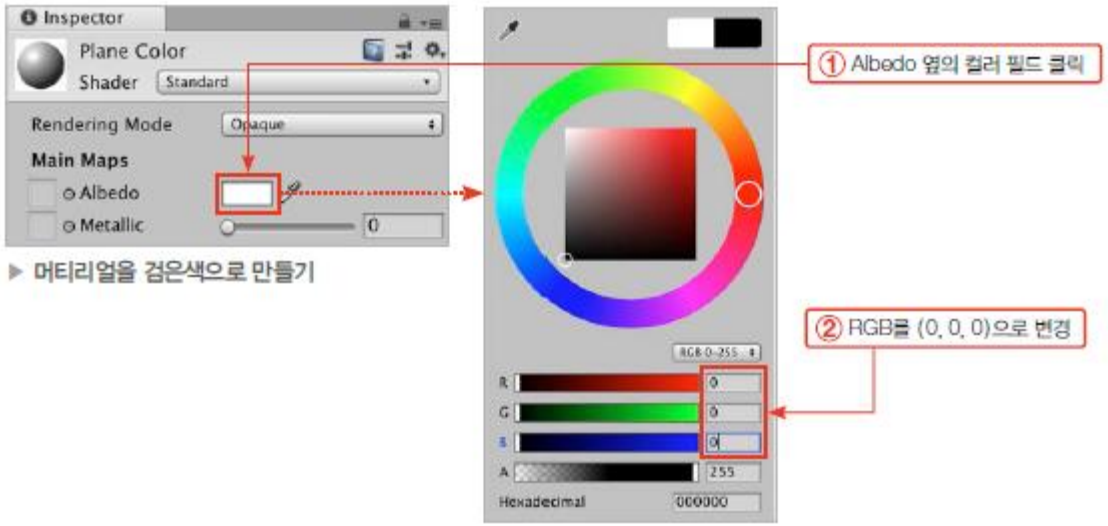
▶ 새로운 머티리얼 만들기

닷지 플레이어 제작

【과정 02】 머티리얼을 검은색으로 만들기

① Albedo 옆의 컬러 필드 클릭

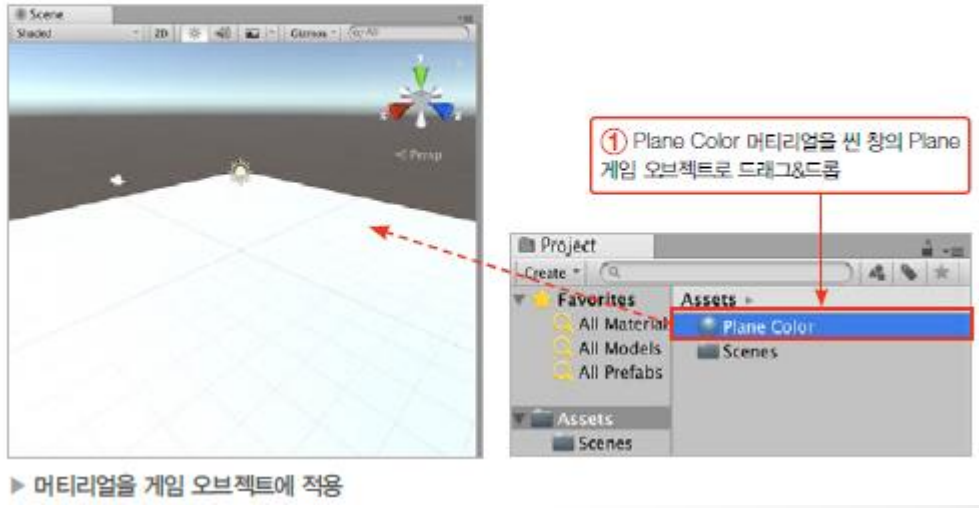
② **컬러** 창에서 RGB 값을 (0, 0, 0)으로 변경 › **컬러** 창 닫기



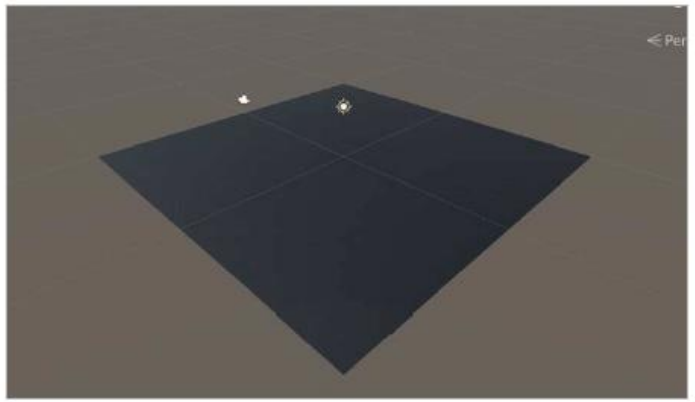
▶ 머티리얼을 검은색으로 만들기

【과정 03】 머티리얼을 게임 오브젝트에 적용

① Plane Color 머티리얼을 씬 창의 Plane 게임 오브젝트로 **드래그&드롭**



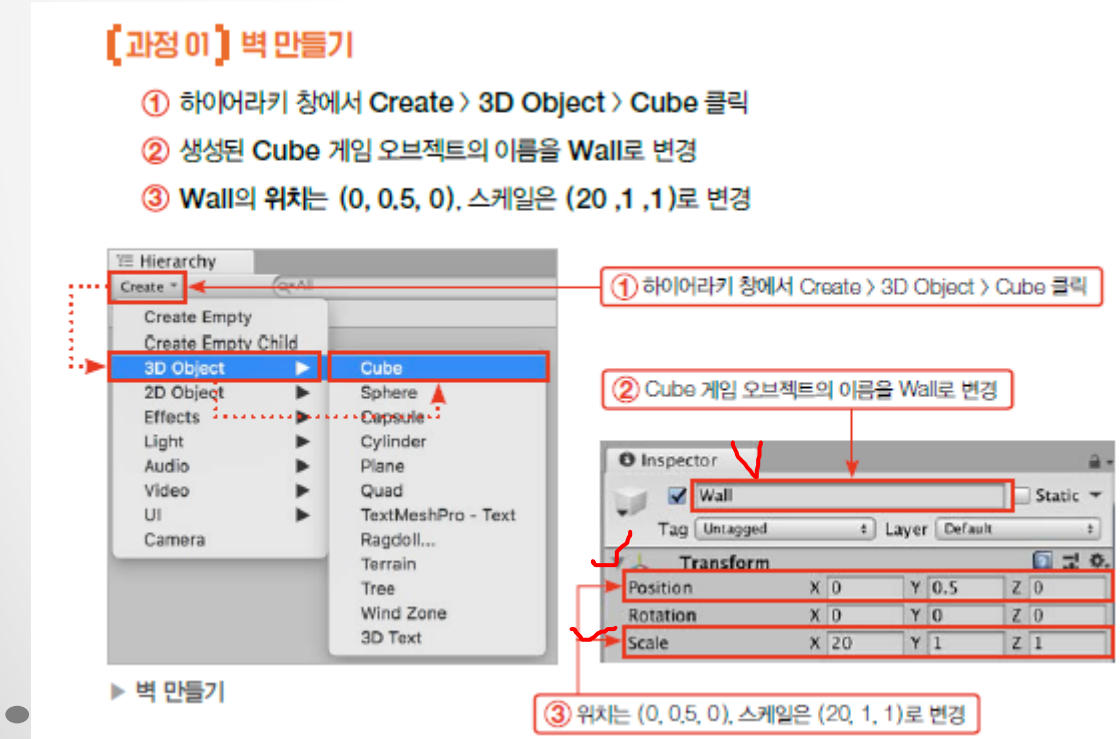
▶ 머티리얼을 게임 오브젝트에 적용

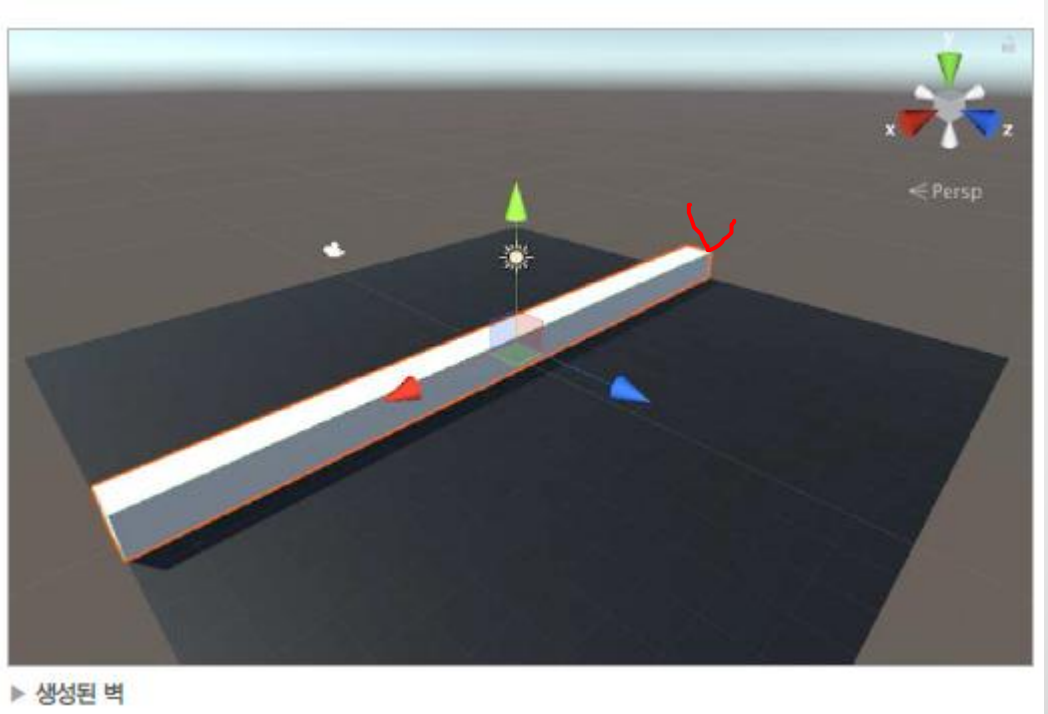


▶ 검은색이 된 Plane 게임 오브젝트

닷지 플레이어 제작

- 벽 만들기
- 게임 도중 플레이어가 바닥을 벗어날 수 없도록 Plane 게임 오브젝트 사방에 벽을 추가함.

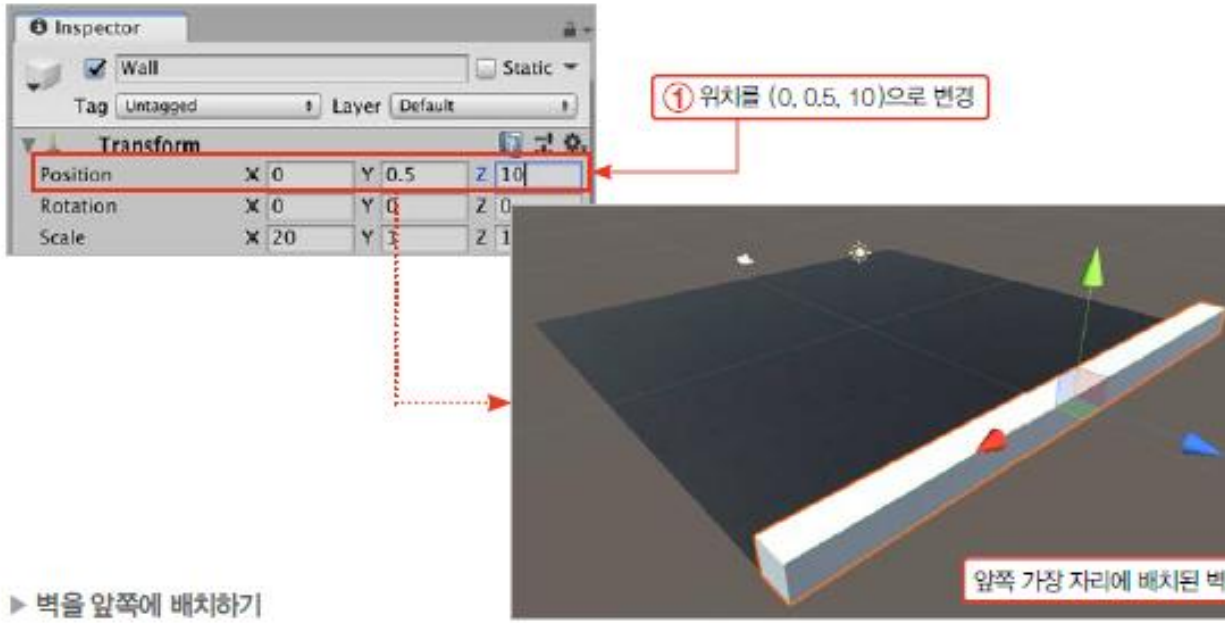




닷지 플레이어 제작

【과정 02】 벽을 앞쪽에 배치하기

① Wall 게임 오브젝트의 **위치**를 (0, 0.5, 10)으로 변경

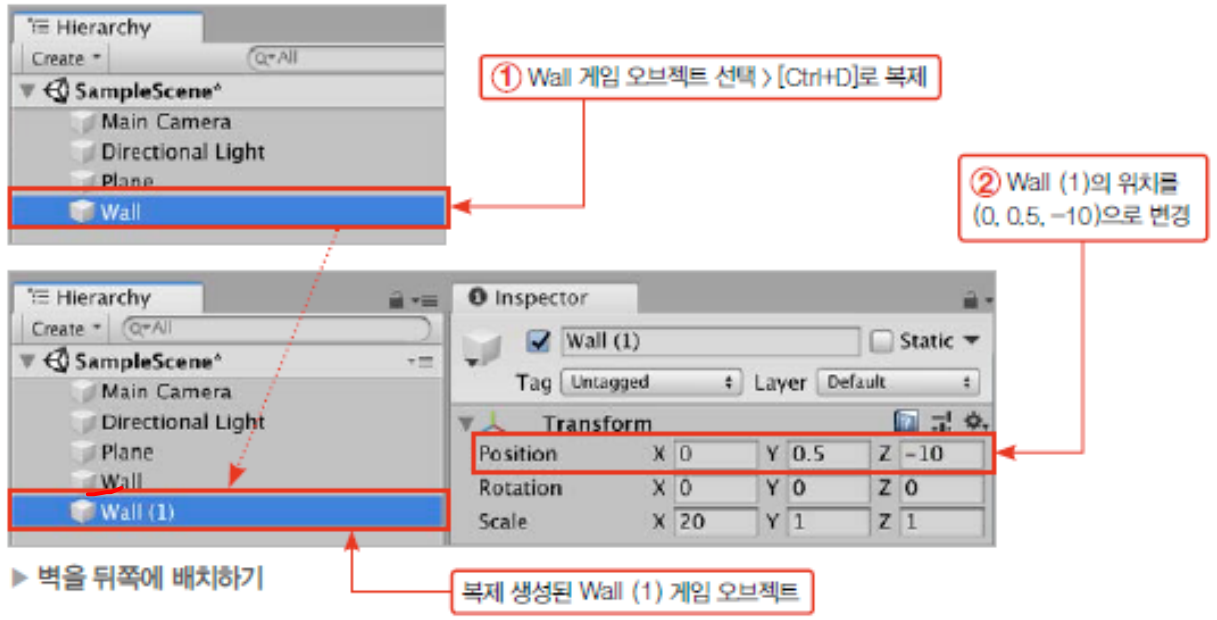


▶ 벽을 앞쪽에 배치하기

【과정 03】 벽을 뒤쪽에 배치하기

① **하이어라키** 창에서 **Wall** 게임 오브젝트 선택 › **[Ctrl+D]**로 복제[1]

② 생성된 **Wall (1)** 게임 오브젝트를 선택하고 **위치**를 (0, 0.5, −10)으로 변경



▶ 벽을 뒤쪽에 배치하기

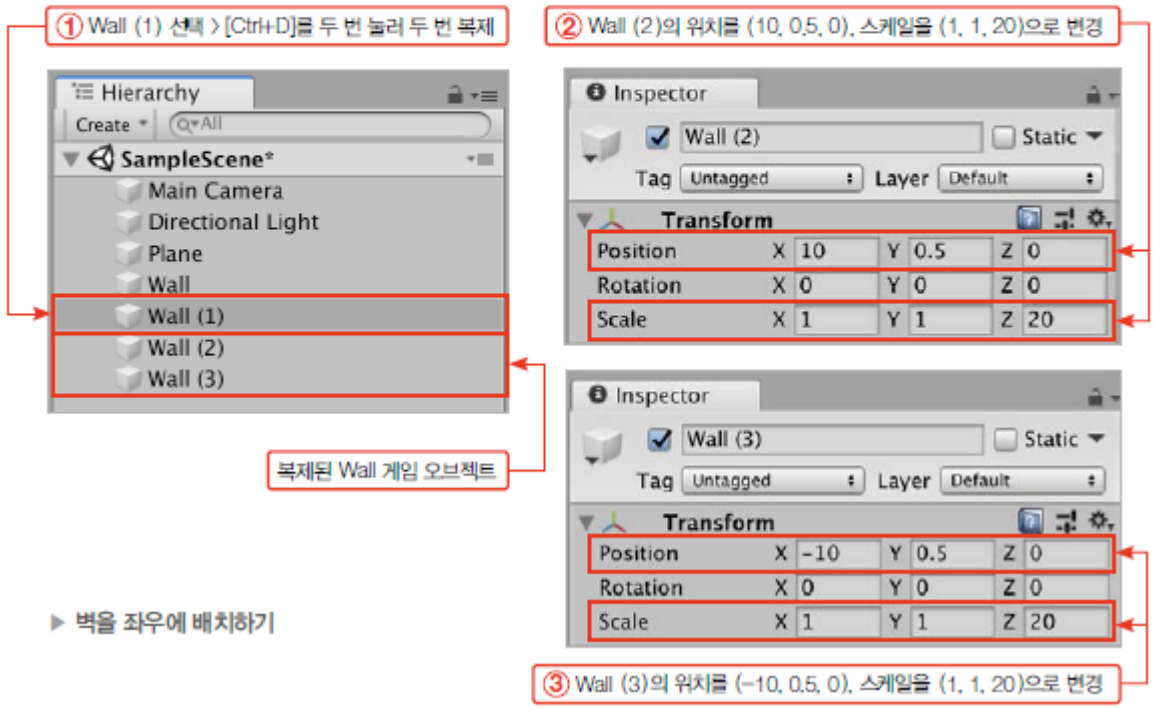
닷지 플레이어 제작

[과정 04] 벽을 좌우에 배치하기

① **하이어라키** 창에서 **Wall (1)** 선택 > **[Ctrl+D]**를 두 번 눌러 두 번 복제

② 생성된 **Wall (2)**의 **위치**를 **(10, 0.5, 0)**, **스케일**을 **(1, 1, 20)**으로 변경

③ 생성된 **Wall (3)**의 **위치**를 **(−10, 0.5, 0)**, **스케일**을 **(1, 1, 20)**으로 변경



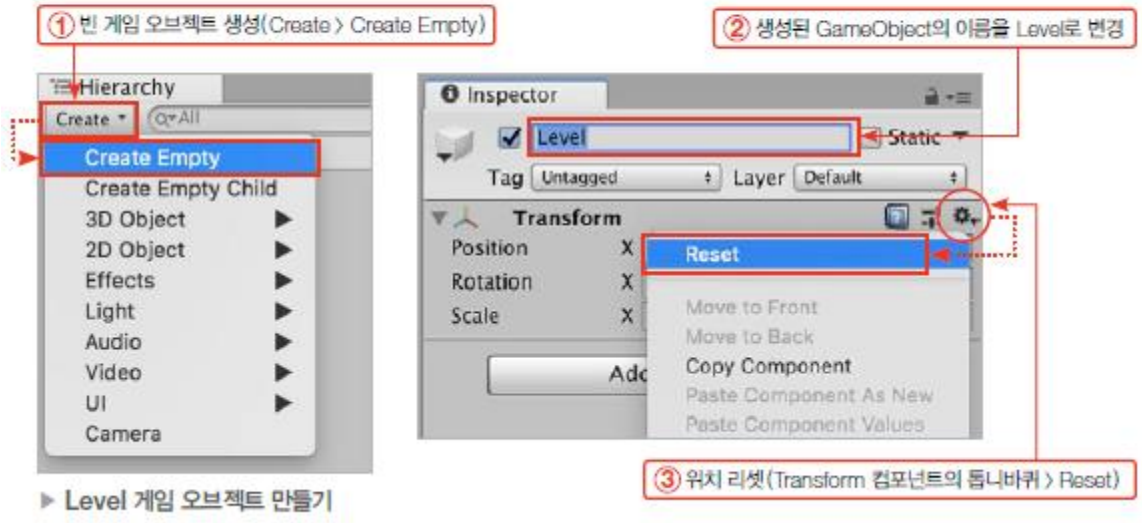
▶ 벽을 좌우에 배치하기

[과정 05] Level 게임 오브젝트 만들기

① **하이어라키** 창에서 **빈 게임 오브젝트** 생성(Create > Create Empty)

② 생성된 GameObject의 이름을 Level로 변경

③ **Level** 게임 오브젝트의 **위치** 리셋(Transform 컴포넌트의 톱니바퀴 > Reset 클릭)



▶ Level 게임 오브젝트 만들기

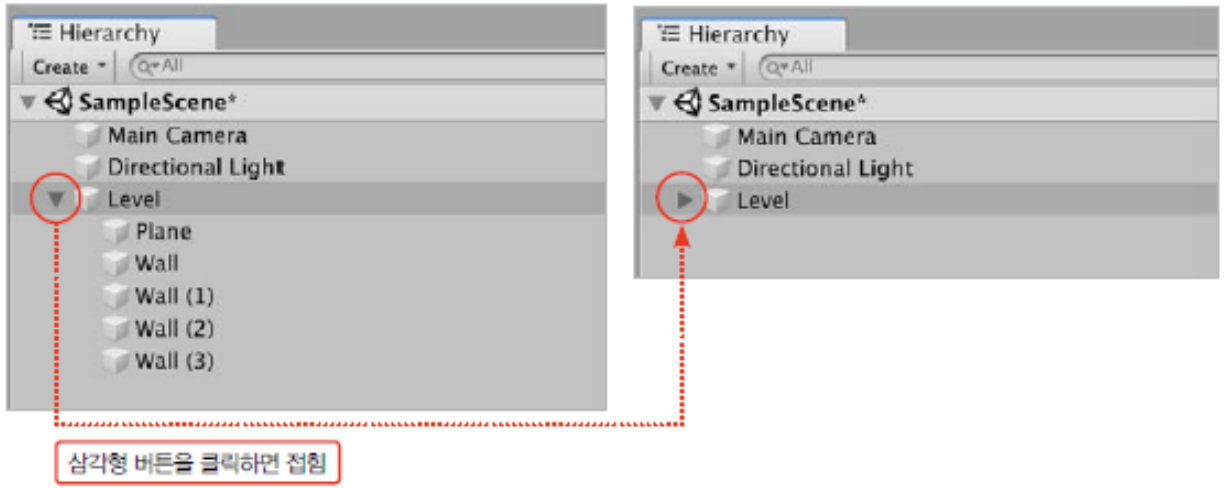
닷지 플레이어 제작

[과정 06] 게임 오브젝트들을 Level의 자식으로 넣기

① **하이어라키** 창에서 **[Shift+클릭]**으로 Plane, Wall, Wall (1), Wall (2), Wall (3)을 모두 선택

② 선택한 게임 오브젝트를 **Level** 게임 오브젝트로 **드래그&드롭**

▶ 게임 오브젝트들을 Level의 자식으로 넣기

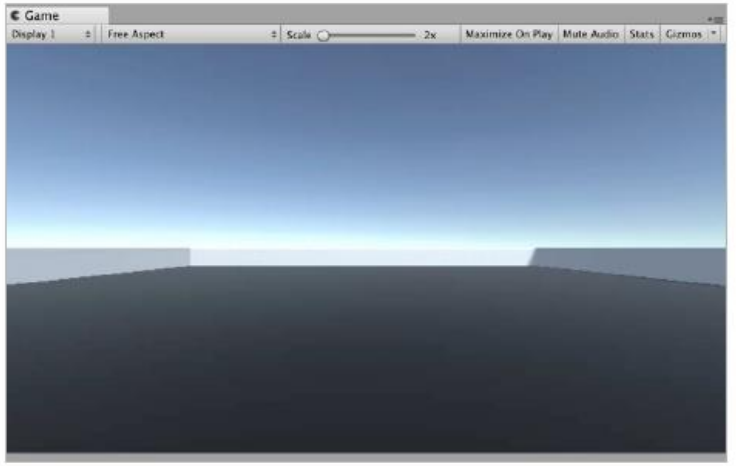


▶ 자식 리스트를 펼치거나 접기

닷지 플레이어 제작

- 게임 창에서 레벨 전체가 한눈에 보이도록 카메라를 배치하겠음

▶ 수정된 카메라가 표시할 게임 화면

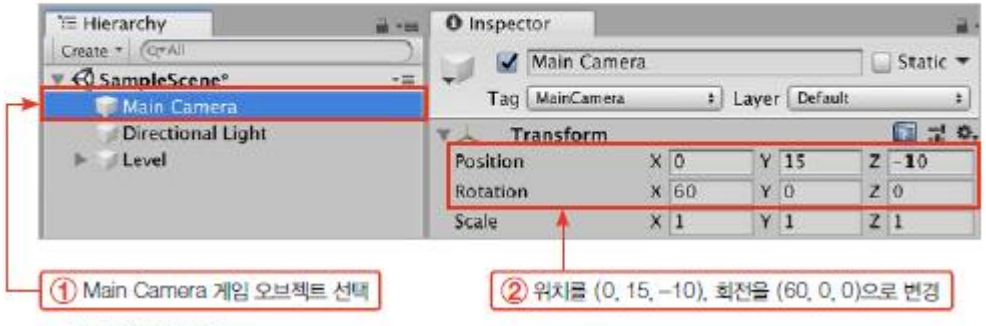


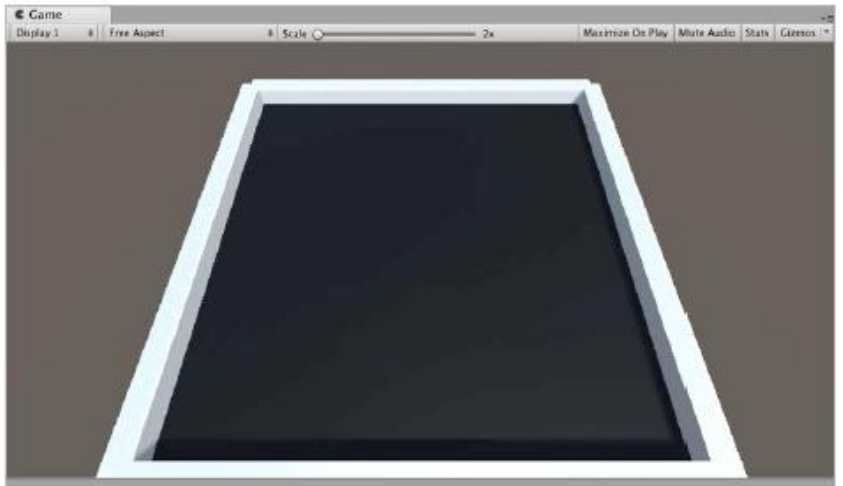
▶ 바닥에 가까이 붙어 있는 카메라

[과정 01] 카메라 위치 변경

① **하이어라키** 창에서 Main Camera 게임 오브젝트 선택

② Main Camera의 **위치**를 (0, 15, −10), **회전**을 (60, 0, 0)으로 변경





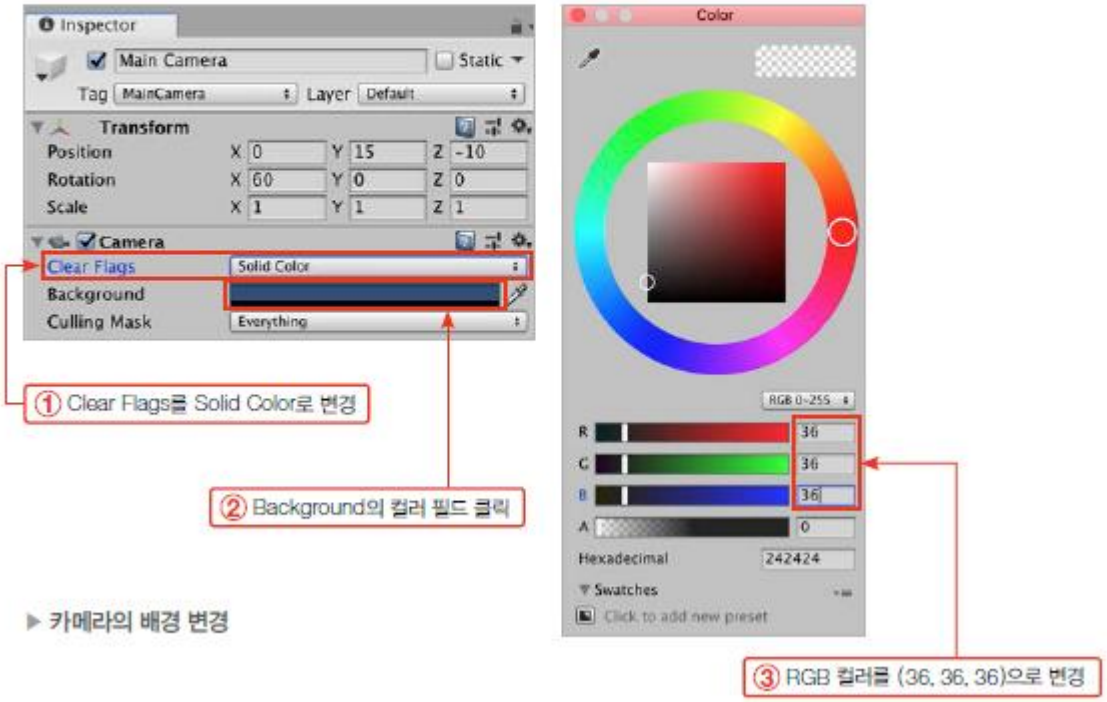
닷지 플레이어 제작

【과정 02】 카메라의 배경 변경

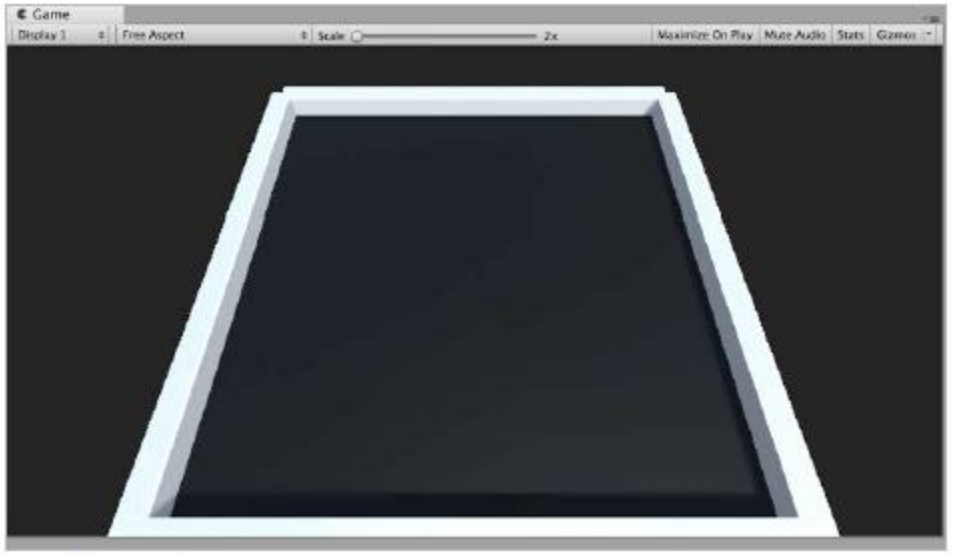
① Main Camera 게임 오브젝트의 Camera 컴포넌트에서 Clear Flags의 값을 Solid Color로 변경

② Background의 컬러 필드 클릭 〉 **컬러** 창 열림

③ RGB 컬러를 (36, 36, 36)으로 변경 〉 **컬러** 창 닫기



▶ 카메라의 배경 변경



▶ 수정된 배경 컬러

【과정 03】 씬 저장하기

① [Ctrl+S]로 씬 저장

닷지 플레이어 제작

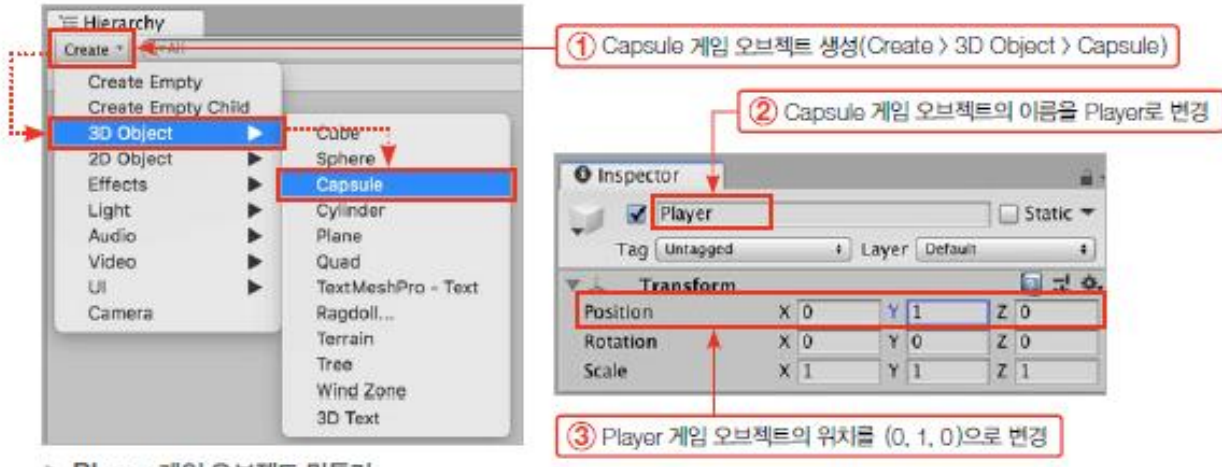
- 플레이어 게임 오브젝트 만들기
- 스크립트를 사용해서 조작할 수 있는 플레이어 게임 오브젝트를 만듬.

[과정 01] Player 게임 오브젝트 만들기

① **하이어라키** 창에서 캡슐(Capsule) 게임 오브젝트 생성(**Create** > **3D Object** > **Capsule**)

② **Capsule** 게임 오브젝트의 **이름**을 **Player**로 변경

③ **Player** 게임 오브젝트의 **위치**를 **(0, 1, 0)**으로 변경

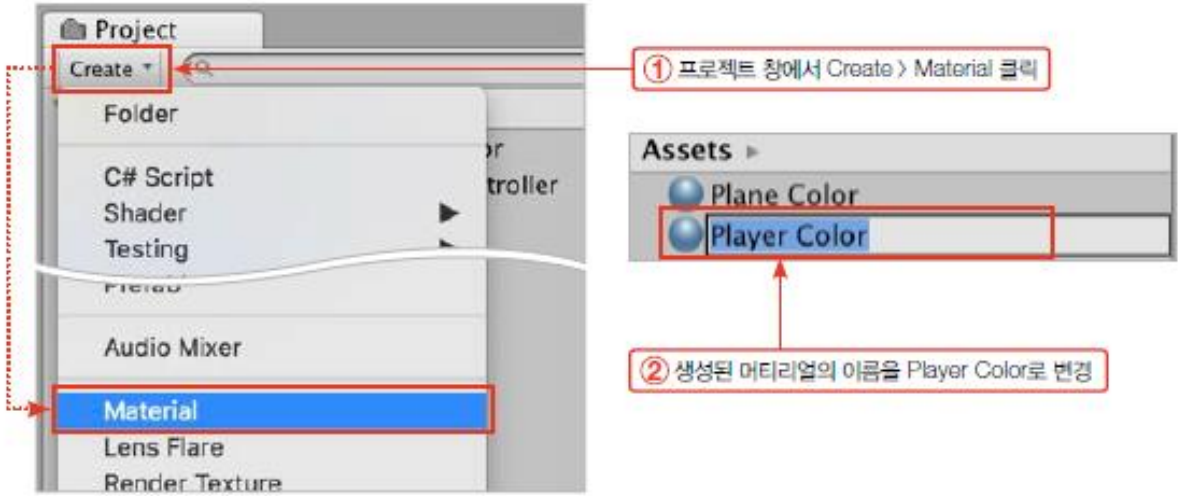


▶ Player 게임 오브젝트 만들기

[과정 02] Player Color 머티리얼 만들기

① **프로젝트** 창에서 **Create** > **Material** 클릭

② 생성된 머티리얼의 이름을 **Player Color**로 변경



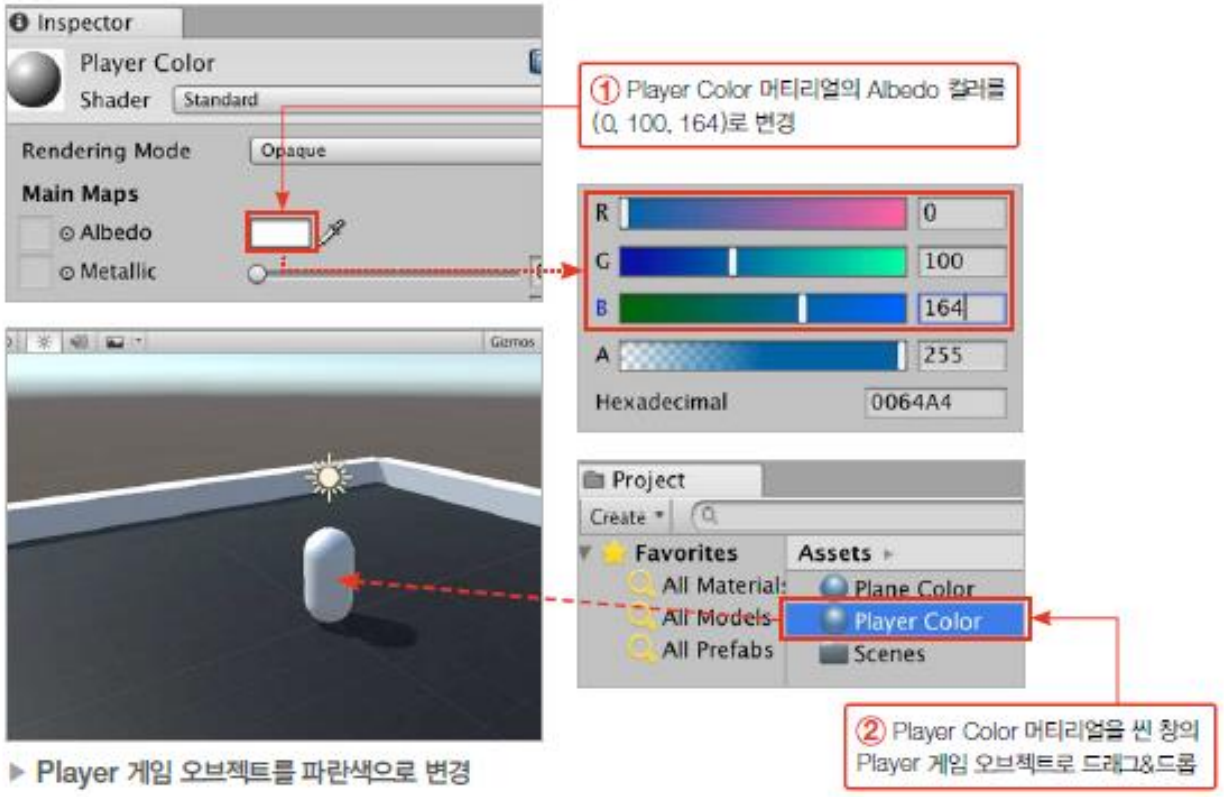
▶ Player Color 머티리얼 만들기

닷지 플레이어 제작

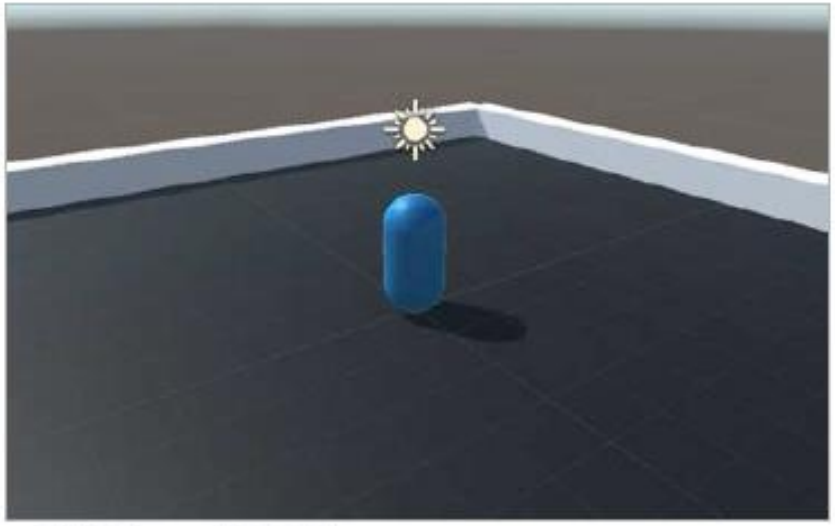
[과정 03] Player 게임 오브젝트를 파란색으로 변경하기

① Player Color 머티리얼의 Albedo 컬러를 (0, 100, 164)로 변경

② Player Color 머티리얼을 씬 창의 Player 게임 오브젝트로 드래그&드롭



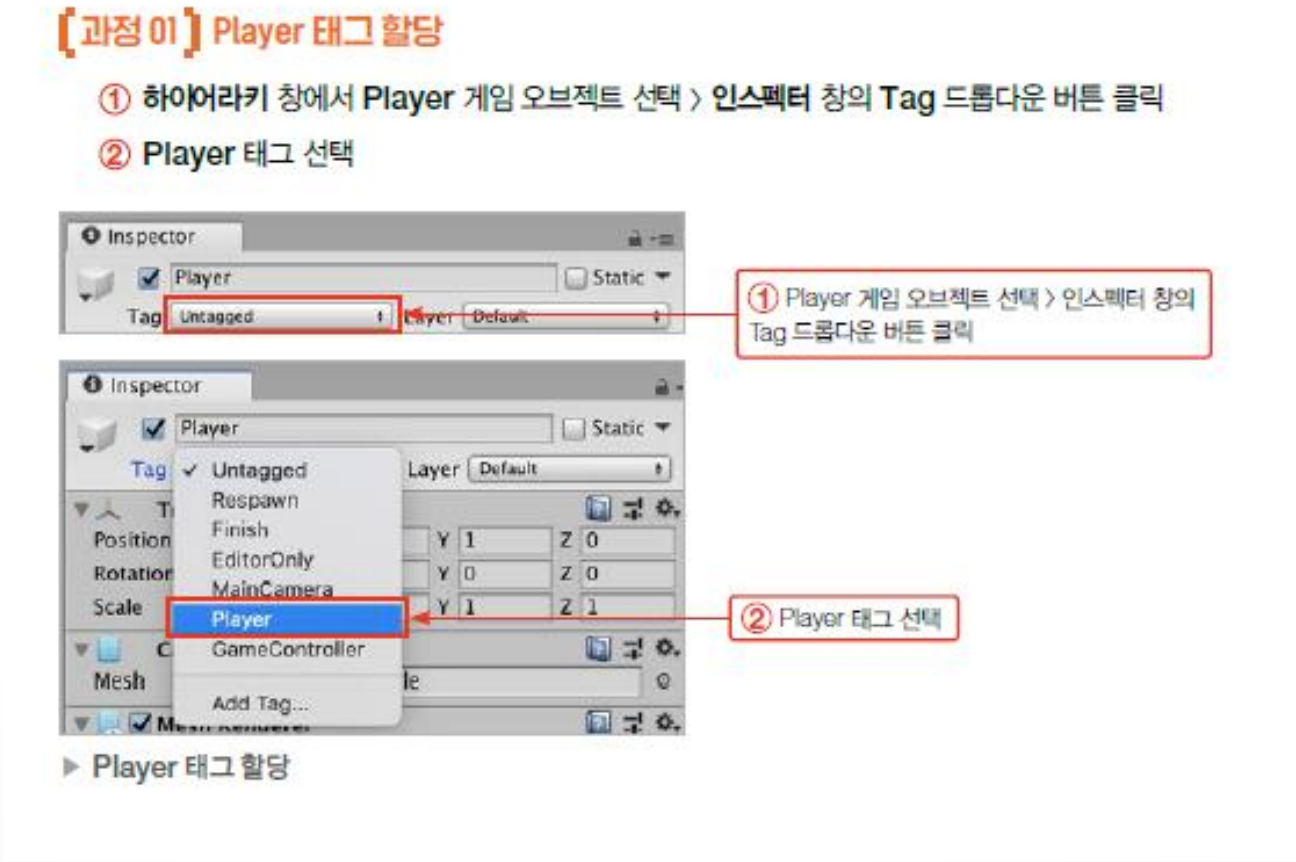
▶ Player 게임 오브젝트를 파란색으로 변경



▶ 완성된 Player 게임 오브젝트

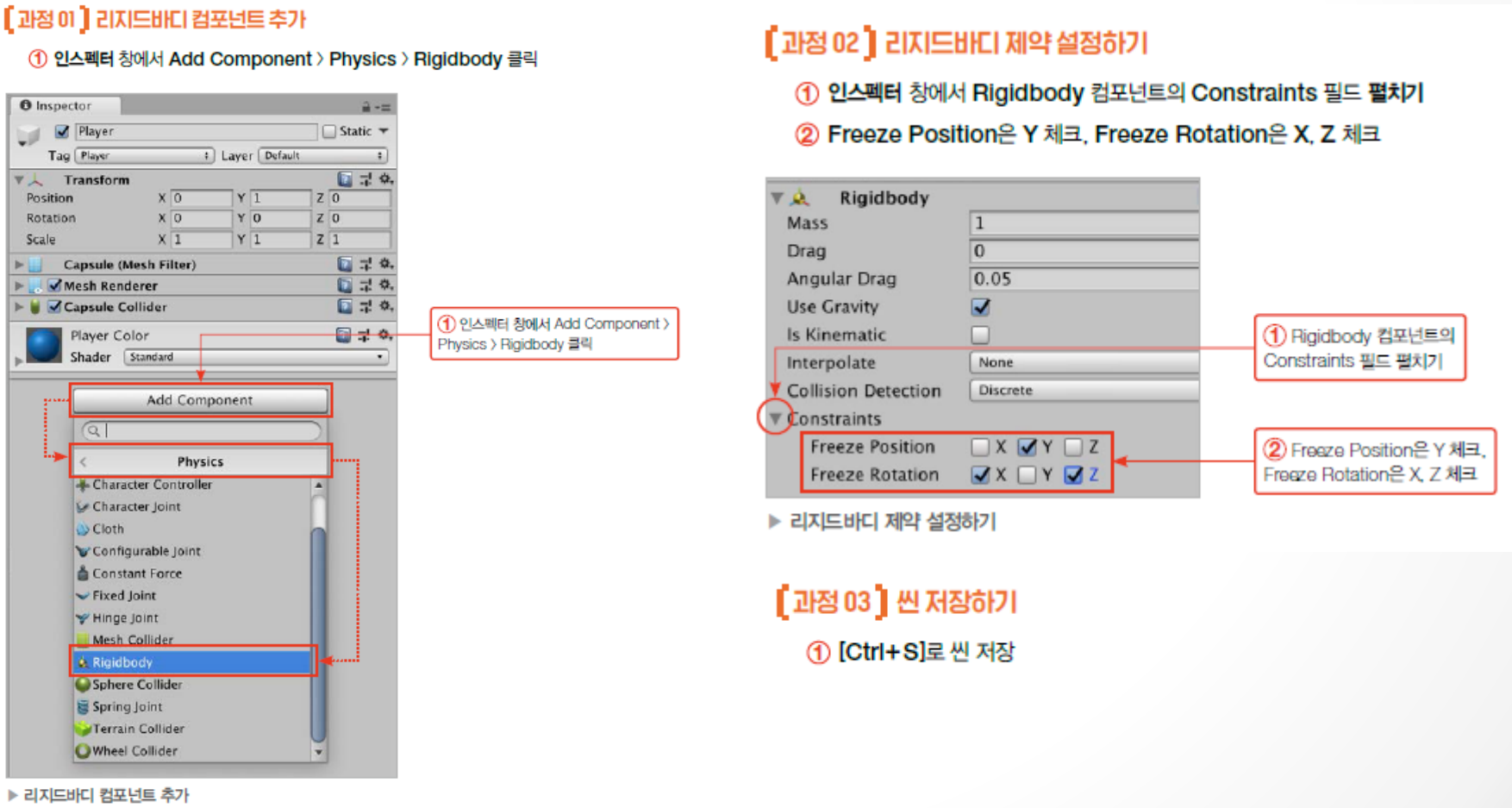
닷지 플레이어 제작

- 태그 설정하기
  - 태그는 게임 오브젝트를 분류하고, 코드에서 게임 오브젝트를 구별하는 데 사용됨



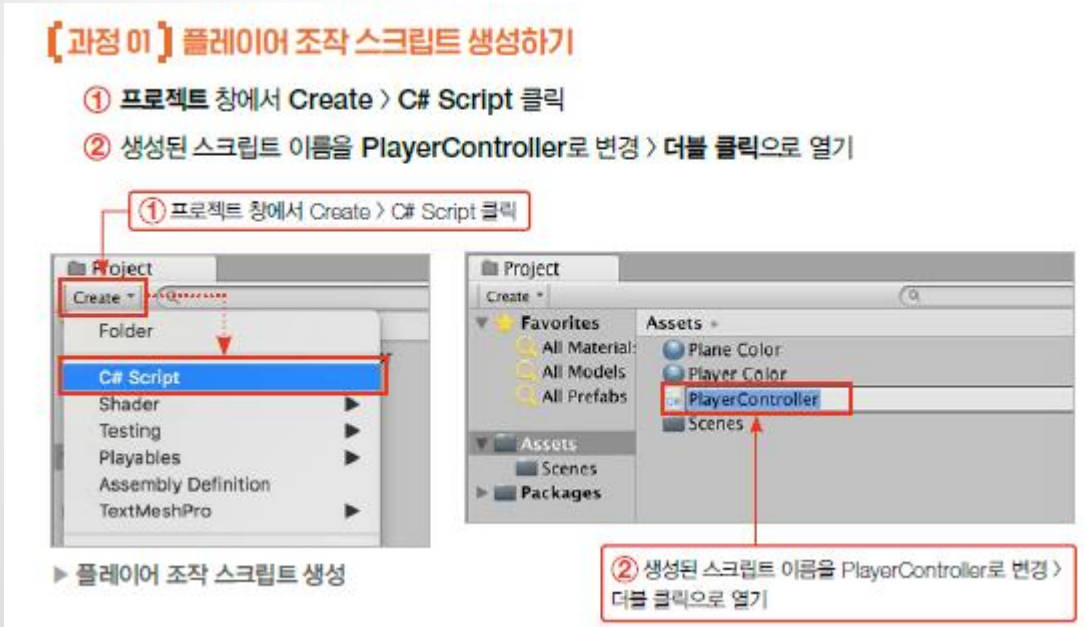
닷지 플레이어 제작

- 리지드바디 컴포넌트 설정
  - Player 게임 오브젝트를 움직이는 데는 물리적인 힘과 속력이 필요함.



닷지 플레이어 제작

- Player 게임 오브젝트를 조종하는 PlayerController 스크립트를 준비함.




```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class PlayerController : MonoBehaviour {

    // Use this for initialization
    void Start() {

    }
```

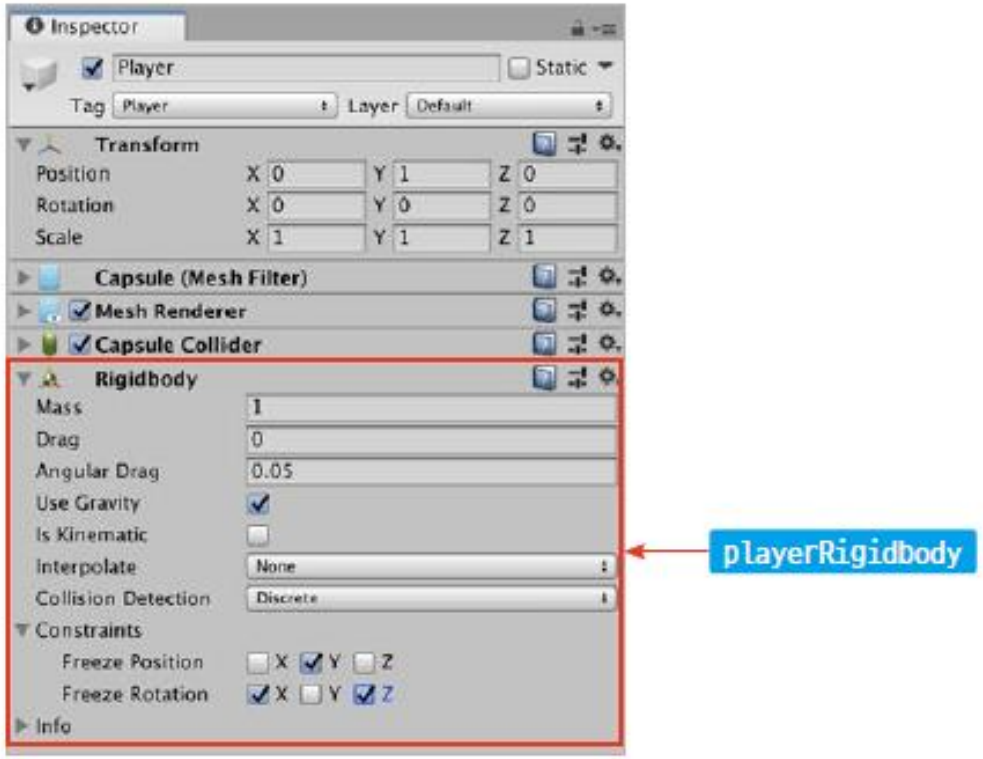
[자동 생성된 기본 코드]

닷지 플레이어 제작

【과정 02】 변수 선언하기

① PlayerController 스크립트를 다음과 같이 수정

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class PlayerController : MonoBehaviour {
    public Rigidbody playerRigidbody; // 이동에 사용할 리지드바디 컴포넌트
    public float speed = 8f; // 이동 속력

    void Start() {

    }

    void Update() {

    }
}
```



▶ playerRigidbody의 용도

닷지 플레이어 제작

- Update( ) 메서드
- 입력을 감지하려면 Update( ) 메서드와 Input 클래스의 입력 감지 메서드가 필요함.

초당 프레임

영화는 1초에 24번, 컴퓨터 화면은 1초에 60번 정도 화면을 새로 그립니다. 매번 새로 그리는 각각의 화면을 프레임Frame이라고 부릅니다.

▶ 주기적으로 새로 그려지는 프레임

Update( ) 메서드

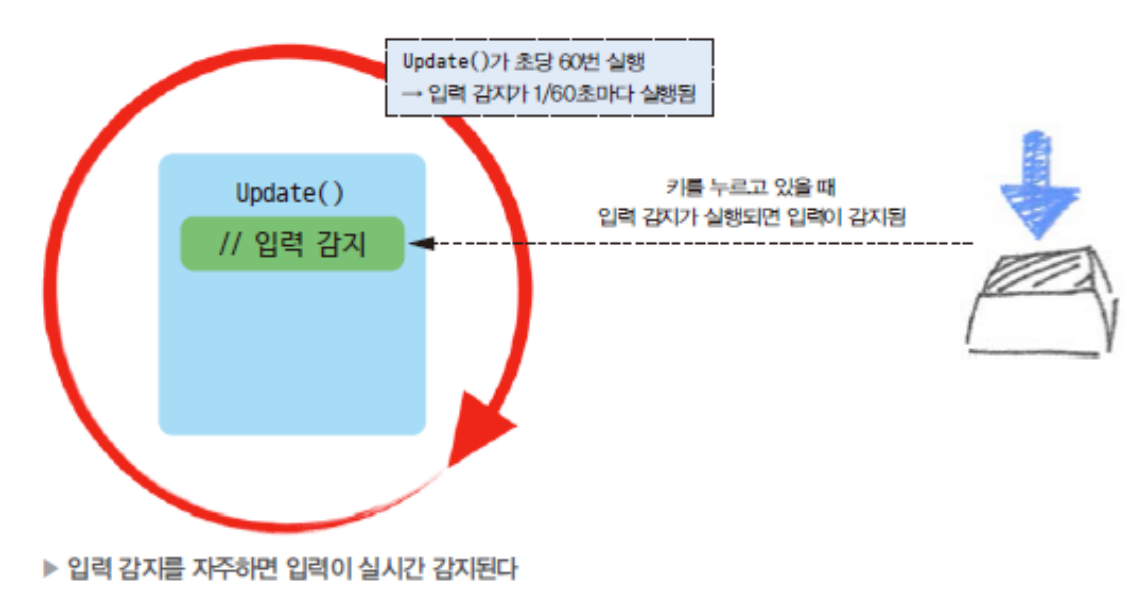
Update( ) 메서드는 Start( ) 메서드처럼 특정 시점에 자동으로 실행되는 유니티 이벤트 메서드입니다. Update( ) 메서드는 한 프레임에 한 번, 매 프레임마다 반복 실행됩니다.

▶ Update( )가 매 프레임마다 실행됨

닷지 플레이어 제작

- Input을 사용한 입력 감지
- 유니티의 Input 클래스는 사용자 입력을 감지하는 메서드를 모아둔 집합임.
- Input의 입력감지 메서드는 실행 시점에 어떤 키를 눌렀는지 알려줌.

닷지 플레이어 제작



【과정 01】 Update() 메서드 작성하기

① PlayerController 스크립트의 Update() 메서드를 다음과 같이 수정

```
void Update() {
    if (Input.GetKey(KeyCode.UpArrow) == true) {
        // 위쪽 방향키 입력이 감지된 경우 z 방향 힘 주기
        playerRigidbody.AddForce(0f, 0f, speed);
    }

    if (Input.GetKey(KeyCode.DownArrow) == true) {
        // 아래쪽 방향키 입력이 감지된 경우 -z 방향 힘 주기
        playerRigidbody.AddForce(0f, 0f, -speed);
    }

    if (Input.GetKey(KeyCode.RightArrow) == true) {
        // 오른쪽 방향키 입력이 감지된 경우 x 방향 힘 주기
        playerRigidbody.AddForce(speed, 0f, 0f);
    }

    if (Input.GetKey(KeyCode.LeftArrow) == true) {
        // 왼쪽 방향키 입력이 감지된 경우 -x 방향 힘 주기
        playerRigidbody.AddForce(-speed, 0f, 0f);
    }
}
```

닷지 플레이어 제작

- Input.GetKey( ) 메서드는 키보드의 식별자를 KeyCode 타입으로 입력받음.

```
bool Input.GetKey(KeyCode key);
```

- Update( ) 내부의 다음 코드도 1초에 수십 번씩 실행됨.

```
if (Input.GetKey(KeyCode.UpArrow) == true) {
    playerRigidbody.AddForce(0f, 0f, speed);
}
```

닷지 플레이어 제작

NOTE_ KeyCode

Input.GetKey ( ) 메서드의 입력으로 사용하는 KeyCode는 키보드의 키 식별자를 쉽게 가리키기 위한 타입입니다. KeyCode 타입은 내부적으로는 숫자로 동작합니다.

키보드의 키에는 식별자가 할당되어 있습니다. 예를 들어 위쪽 방향키의 식별자는 273입니다. 하지만 숫자로 된 키 식별자를 모두 외우는 것은 무리입니다. 따라서 키 식별자 273에 대응하는 KeyCode.UpArrow를 사용합니다.

NOTE_ Input.GetKey( ) 계열 메서드

Input.GetKey ( ) 메서드는 지정한 키를 누르는 동안 true를 반환합니다. 이외에도 Input.GetKey ( ) 메서드처럼 키보드 입력을 감지하지만, 감지 시점이 다른 Input.GetKey ( ) 계열의 메서드들이 있습니다.

- Input.GetKey ( ) : 해당 키를 '누르는 동안' true, 그 외에는 false 반환
- Input.GetKeyDown ( ) : 해당 키를 '누르는 순간' true, 그 외에는 false 반환

닷지 플레이어 제작

- Die( ) 메서드는 자신의 게임 오브젝트를 비활성화하여 '죽음'을 구현하는 메서드임.

【과정 01】 PlayerController 스크립트에 Die( ) 메서드 추가

① Update( ) 메서드 아래에 다음과 같이 Die( ) 메서드 추가

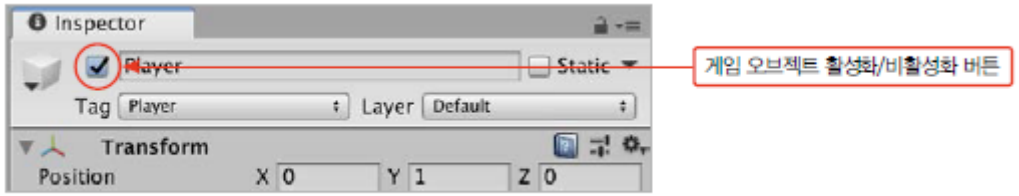
```
// using 문 생략

public class PlayerController : MonoBehaviour {
    public Rigidbody playerRigidbody; // 이동에 사용할 리지드바디 컴포넌트
    public float speed = 8f; // 이동 속력

    void Start() {

    }

    void Update() {
        // Update() 메서드 내용 생략
    }

    public void Die() {
        // 자신의 게임 오브젝트를 비활성화
        gameObject.SetActive(false);
    }
}
```

닷지 플레이어 제작

- gameObject는 컴포넌트 입장에서 자신이 추가된 게임 오브젝트를 가리키는 변수임.
- gameObject는 GameObject 타입의 변수이며 컴포넌트들의 기반 클래스인 MonoBehaviour에서 제공함.

닷지 플레이어 제작

- 모든 게임 오브젝트는 스스로를 끄고 켜는 기능을 가지고 있음.



▶ 게임 오브젝트 활성화/비활성화 버튼

- 게임 오브젝트 활성화/비활성화를 코드 상에서는 SetActive( ) 메서드로 실행할 수 있음.

```
void SetActive(bool value);
```

- gameObject.SetActive(false);는 다음과 같은 순서로 자신의 게임 오브젝트를 비활성화함.
-

1. gameObject를 사용해 자신의 게임 오브젝트에 접근
2. 접근한 게임 오브젝트의 SetActive(false);를 실행

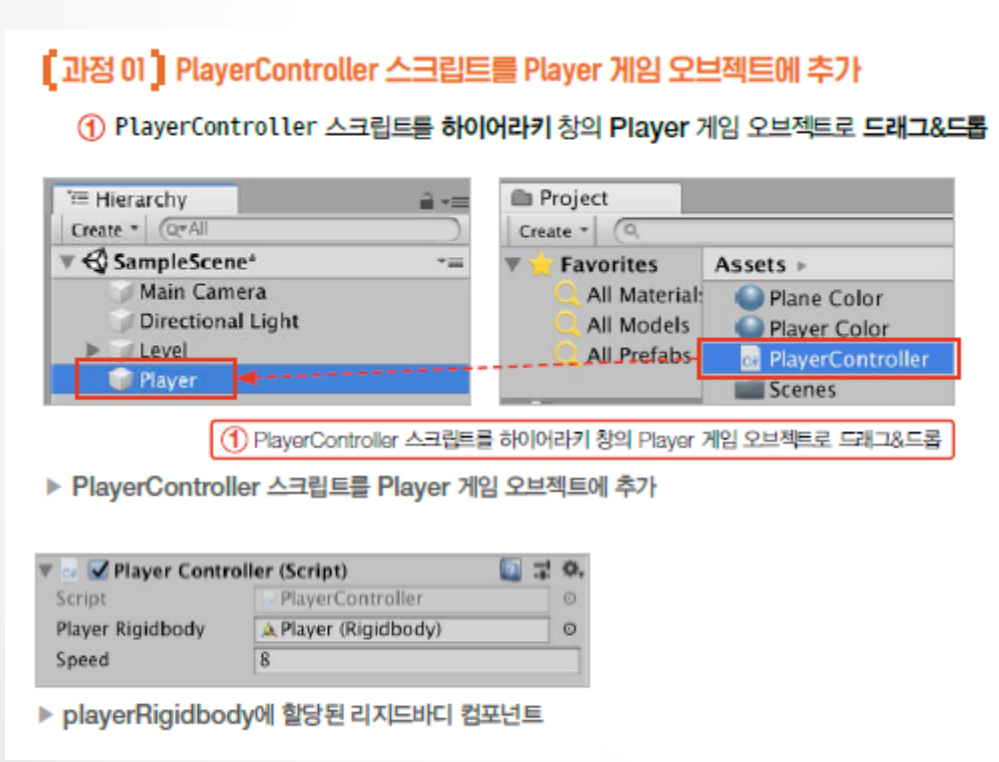
닷지 볼

- 완성된 스크립트 확인(1차)
- Ctrl + S로 꼭 저장하기!

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class PlayerController : MonoBehaviour {
    public Rigidbody playerRigidbody; // 이동에 사용할 리지드바디 컴포넌트
    public float speed = 8f; // 이동 속력

    void Start() {
    }

    void Update() {
        if (Input.GetKey(KeyCode.UpArrow) == true) {
            // 위쪽 방향키 입력이 감지된 경우 z 방향 힘 주기
            playerRigidbody.AddForce(0f, 0f, speed);
        }

        if (Input.GetKey(KeyCode.DownArrow) == true) {
            // 아래쪽 방향키 입력이 감지된 경우 -z 방향 힘 주기
            playerRigidbody.AddForce(0f, 0f, -speed);
        }

        if (Input.GetKey(KeyCode.RightArrow) == true) {
            // 오른쪽 방향키 입력이 감지된 경우 x 방향 힘 주기
            playerRigidbody.AddForce(speed, 0f, 0f);
        }

        if (Input.GetKey(KeyCode.LeftArrow) == true) {
            // 왼쪽 방향키 입력이 감지된 경우 -x 방향 힘 주기
            playerRigidbody.AddForce(-speed, 0f, 0f);
        }
    }

      public void Die() {
          // 자신의 게임 오브젝트를 비활성화
          gameObject.SetActive(false);
      }
  }
```

닷지 플레이어 제작

- PlayerController 컴포넌트 설정하기
  - 완성한 PlayerController 스크립트를 Player 게임 오브젝트에 컴포넌트로 추가하고 실행함.



닷지 플레이어 제작

【과정 03】 테스트하기

① **플레이 버튼** 클릭 > 씬이 시작됨

② **키보드 방향키**로 플레이어 조작하기

【과정 04】 씬 저장하기

① **플레이 버튼** 클릭 > 플레이 모드 해제

② [Ctrl+S]로 씬 저장