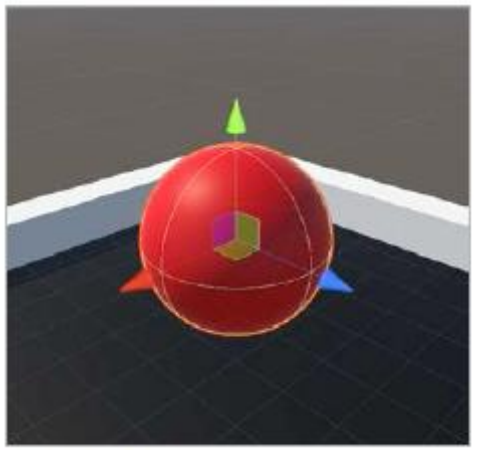
유니티 프로그래밍 강의

이준

닷지 탄알 제작

7.1 탄알 게임 오브젝트 준비

- 7.1.1 Bullet 게임 오브젝트 만들기
  - 탄알 게임 오브젝트를 만들고 필요한 컴포넌트를 추가함.
  - 탄알이 될 Bullet 게임 오브젝트를 구성함. 구Sphere를 생성하고 크기를 적당히 줄여 탄알로 사용함.



▶ 탄알의 모습

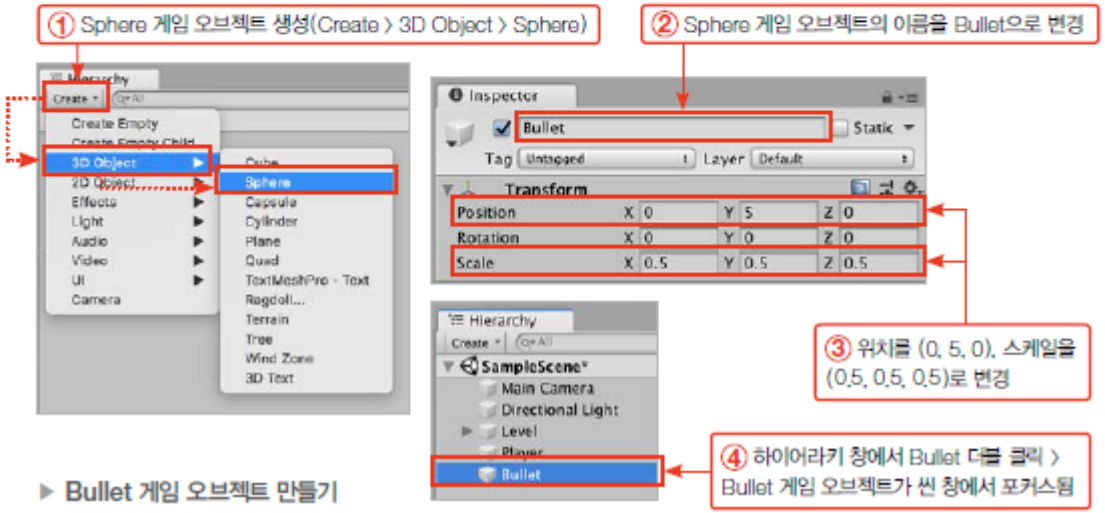
[과정 01] Bullet 게임 오브젝트 생성

① 구(Sphere) 게임 오브젝트 생성(**하이어라키** 창에서 **Create** > **3D Object** > **Sphere**)

② **Sphere** 게임 오브젝트의 이름을 **Bullet**으로 변경

③ **Bullet** 게임 오브젝트의 **위치**를 (0, 5, 0), **스케일**을 (0.5, 0.5, 0.5)로 변경

④ **하이어라키** 창에서 **Bullet** 더블 클릭 > **Bullet** 게임 오브젝트가 **씬** 창에서 포커스됨



▶ Bullet 게임 오브젝트 만들기

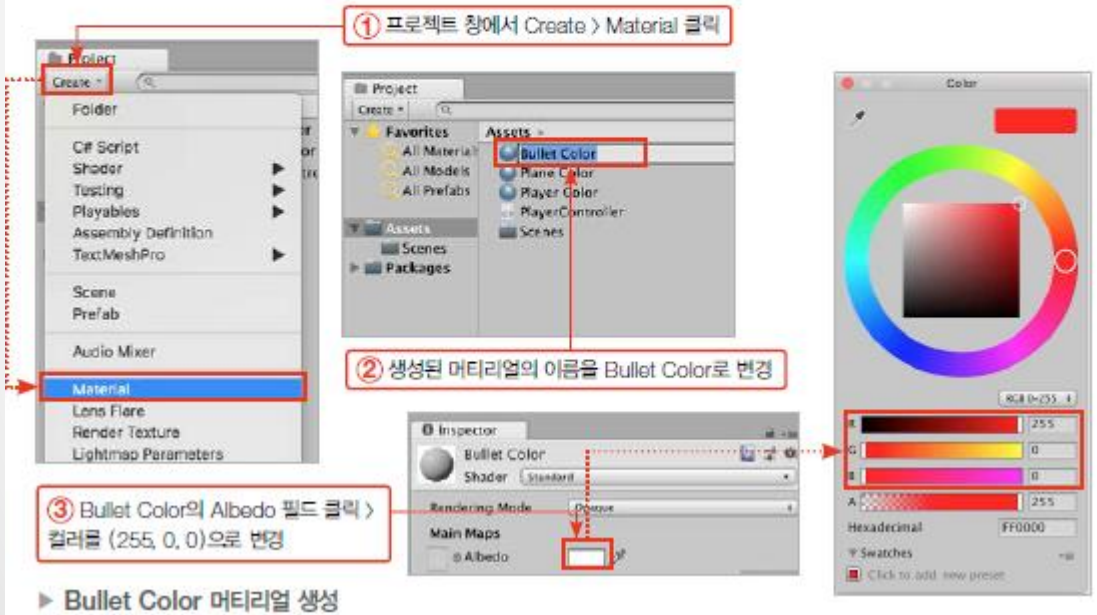
7.1 탄알 게임 오브젝트 준비

[과정 02] Bullet Color 머티리얼 생성

① **프로젝트** 창에서 Create > Material 클릭

② 생성된 머티리얼의 **이름**을 Bullet Color로 변경

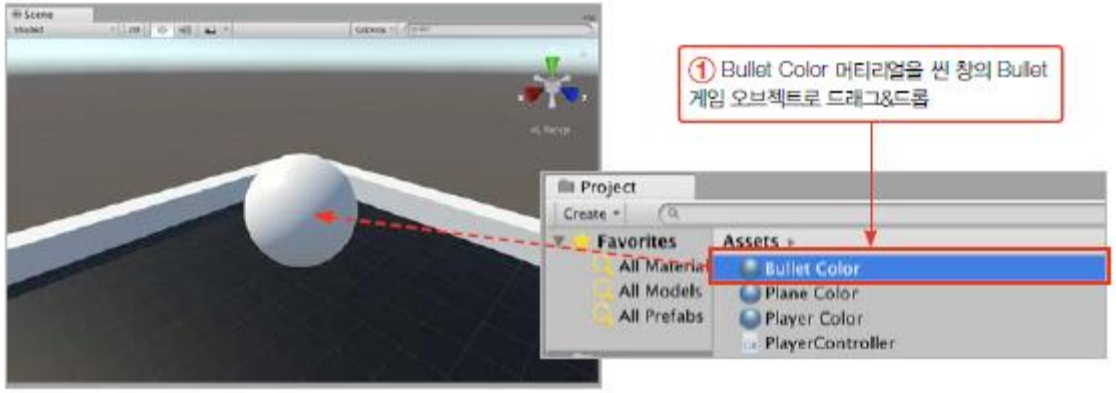
③ **인스펙터** 창에서 Bullet Color의 Albedo 필드 클릭 > **컬러**를 (255, 0, 0)으로 변경



▶ Bullet Color 머티리얼 생성

[과정 03] Bullet 게임 오브젝트를 붉은색으로 변경

① **프로젝트** 창의 Bullet Color 머티리얼을 **씬** 창의 Bullet 게임 오브젝트로 **드래그&드롭**



▶ Bullet 게임 오브젝트를 붉은색으로 변경

7.1 탄알 게임 오브젝트 준비

- 7.1.2 리지드바디 컴포넌트 설정하기

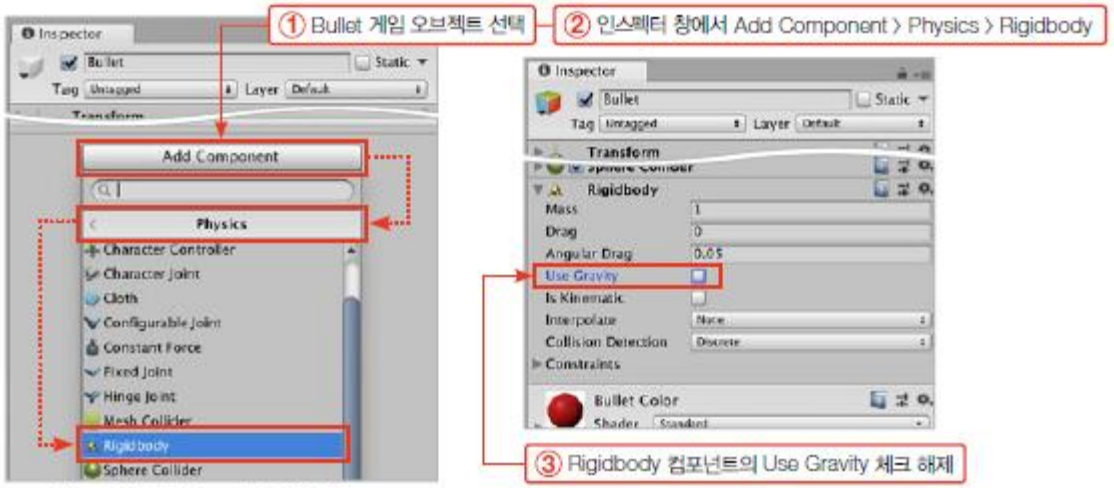
  - 탄알이 속도를 가지도록 Bullet 게임 오브젝트에 리지드바디 컴포넌트를 추가함.

[과정 01] 리지드바디 컴포넌트 추가 및 설정

① **하이어라키** 창에서 **Bullet** 게임 오브젝트 선택

② **인스펙터** 창에서 **Add Component** > **Physics** > **Rigidbody** 클릭

③ 추가된 **Rigidbody** 컴포넌트의 **Use Gravity** 체크 해제



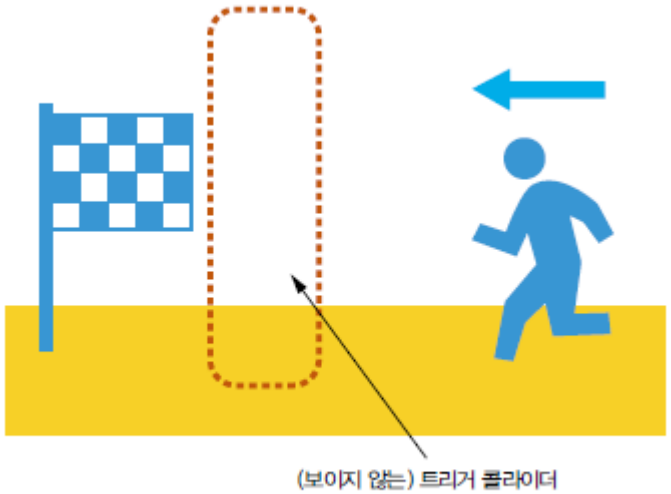
▶ 리지드바디 컴포넌트 추가 및 설정

7.1 탄알 게임 오브젝트 준비

- 7.1.3 콜라이더 설정하기

  - Bullet 게임 오브젝트에는 구 콜라이더Sphere Collider가 추가되어 있어 물리적인 표면이 존재함
  - 따라서 Bullet 게임 오브젝트는 콜라이더를 가진 다른 게임 오브젝트와 충돌하면 튕겨나갈 수 있음.

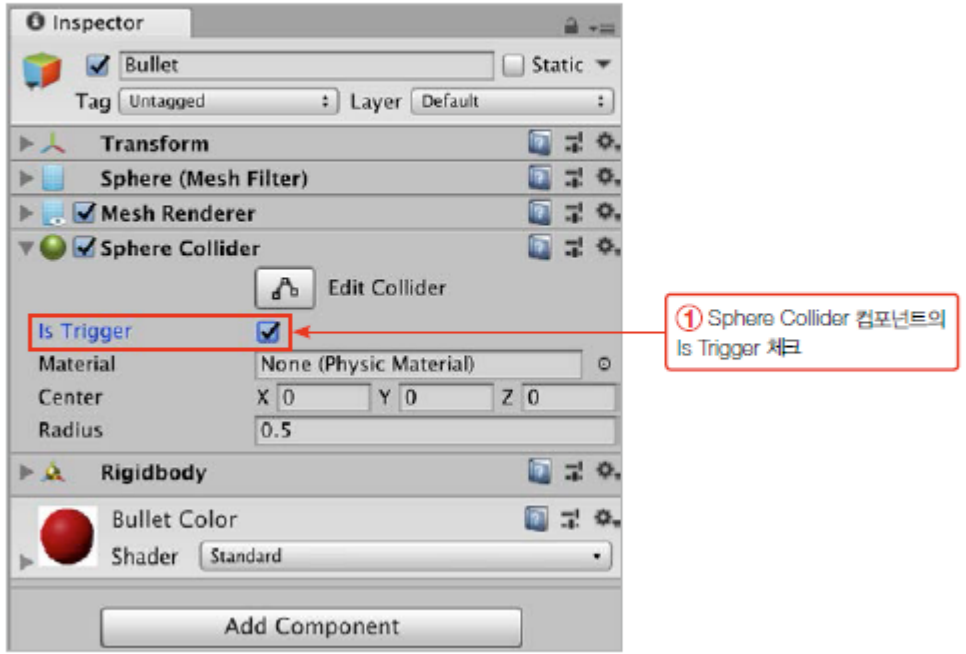
트리거 콜라이더



▶ 트리거 콜라이더 활용법

[과정 01] Bullet 게임 오브젝트의 콜라이더를 트리거로 만들기

① 인스펙터 창에서 Sphere Collider 컴포넌트의 Is Trigger 체크

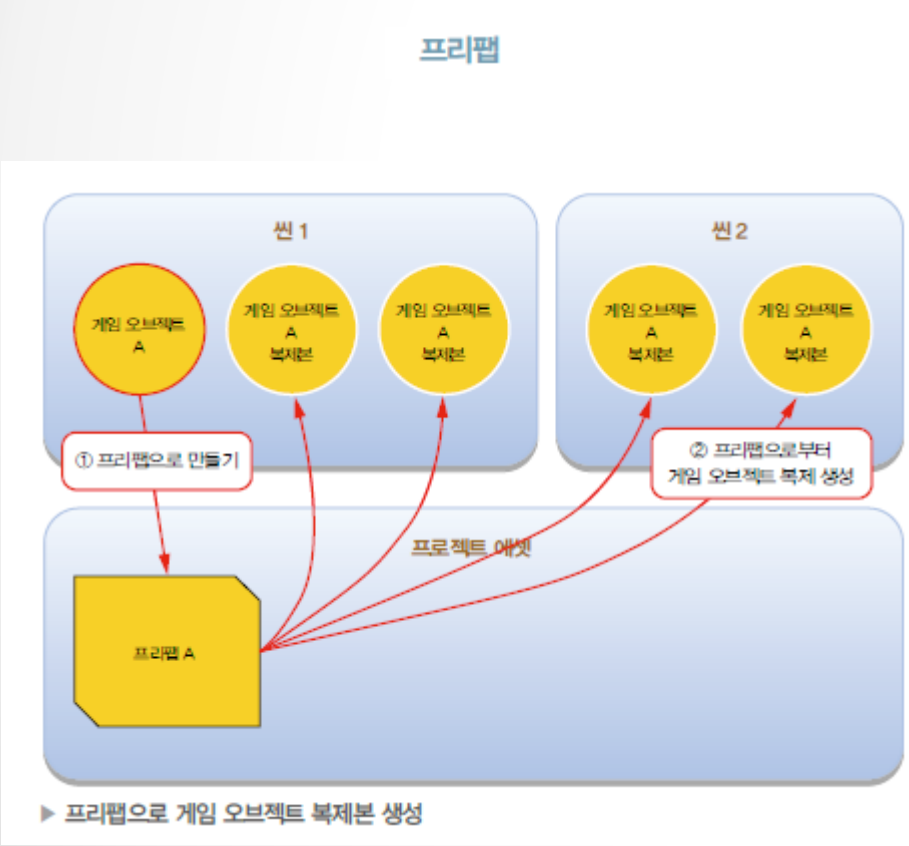


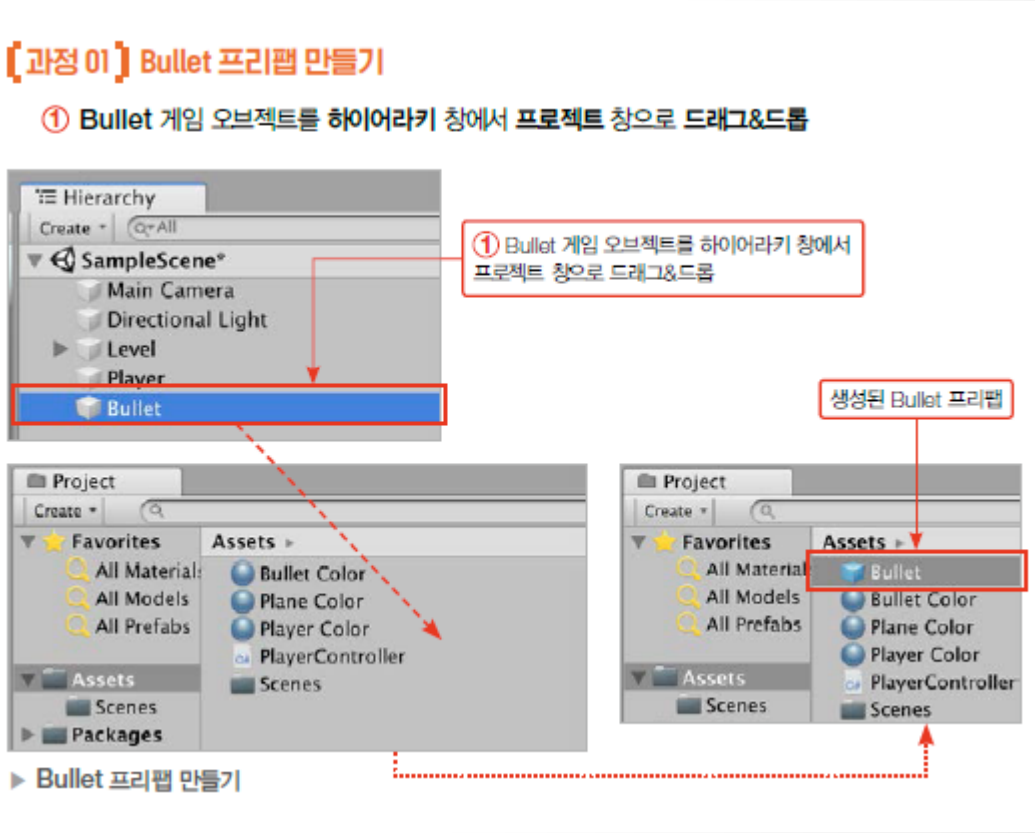
▶ Bullet 게임 오브젝트의 콜라이더를 트리거로 만들기

7.1 탄알 게임 오브젝트 준비

- 7.1.4 Bullet을 프리팹으로 만들기

  - 게임 Bullet 게임 오브젝트를 프리팹Prefab으로 만들겠음.

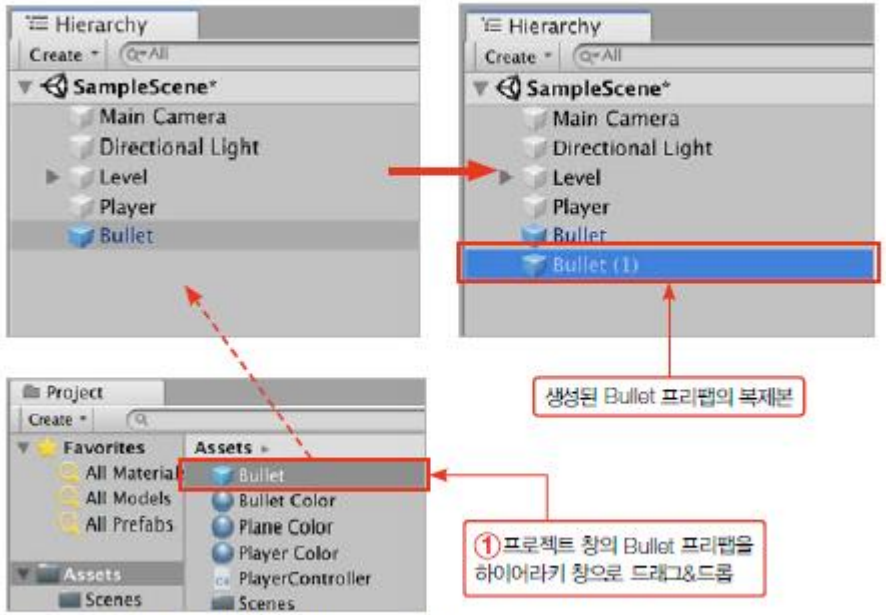




7.1 탄알 게임 오브젝트 준비

[과정 02] Bullet 프리팹에서 Bullet 게임 오브젝트 생성

① **프로젝트** 창의 **Bullet** 프리팹을 **하이어라키** 창으로 **드래그&드롭**



▶ Bullet 프리팹에서 Bullet 게임 오브젝트 생성

[과정 03] Bullet (1) 게임 오브젝트 삭제하기

① **인스펙터** 창에서 **Bullet (1)** 선택

② **[Delete]** 키를 눌러 삭제(맥OS는 **[command+delete]**).

7.2 탄알 스크립트 준비

◦ 7.2.1 Bullet의 변수

- 탄알이 실제로 동작할 수 있도록 Bullet 스크립트를 생성함.
- Bullet 스크립트에 필요한 변수는 다음과 같음.

[과정 01] Bullet 스크립트 생성

① **프로젝트** 창에서 **Create** > **C# Script** 클릭
② 생성된 C# 스크립트의 **이름**을 **Bullet**으로 변경
③ **Bullet** 스크립트를 **더블 클릭**하여 **열기**

- **speed** : 이동 속력
- **bulletRigidbody** : Bullet 게임 오브젝트의 리지드바디 컴포넌트

[과정 01] Bullet에 변수 선언

① **Bullet** 스크립트를 다음과 같이 수정

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

public class Bullet : MonoBehaviour {
    public float speed = 8f; // 탄알 이동 속력
    private Rigidbody bulletRigidbody; // 이동에 사용할 리지드바디 컴포넌트

    void Start() {

    }
}
```

7.2 탄알 스크립트 준비

- 7.2.2 탄알 속도 지정하기

  - bulletRigidbody.velocity로 탄알의 속도를 변경함.

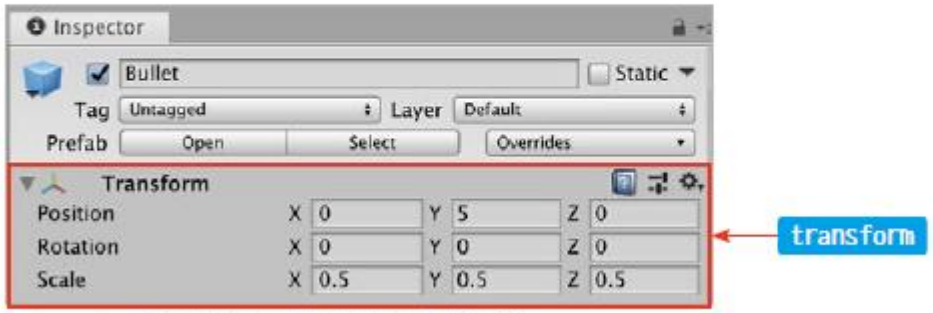
[과정 01] Start() 메서드에서 탄알 속도 지정하기

① Bullet 스크립트의 Start() 메서드를 다음과 같이 수정

```
void Start() {
    // 게임 오브젝트에서 Rigidbody 컴포넌트를 찾아 bulletRigidbody에 할당
    bulletRigidbody = GetComponent<Rigidbody>();
    // 리지드바디의 속도 = 앞쪽 방향 * 이동 속력
    bulletRigidbody.velocity = transform.forward * speed;
}
```

transform

Transform 타입의 변수 transform은 자신의 게임 오브젝트의 트랜스폼 컴포넌트로 바로 접근하는 변수입니다.



▶ transform은 자신의 트랜스폼 컴포넌트를 가리킨다

7.2 탄알 스크립트 준비

- 7.2.3 탄알 자동 파괴하기

  - 탄알이 생성된 후 일정 시간이 흐르면 탄알이 스스로 자동 파괴되게 함.

[과정 01] 탄알 자동 파괴 기능 추가

① Bullet 스크립트의 Start() 메서드를 다음과 같이 수정

```
void Start() {
    // 게임 오브젝트에서 Rigidbody 컴포넌트를 찾아 bulletRigidbody에 할당
    bulletRigidbody = GetComponent<Rigidbody>();
    // 리지드바디의 속도 = 앞쪽 방향 * 이동 속력
    bulletRigidbody.velocity = transform.forward * speed;

    // 3초 뒤에 자신의 게임 오브젝트 파괴
    Destroy(gameObject, 3f);
}
```

Destroy()

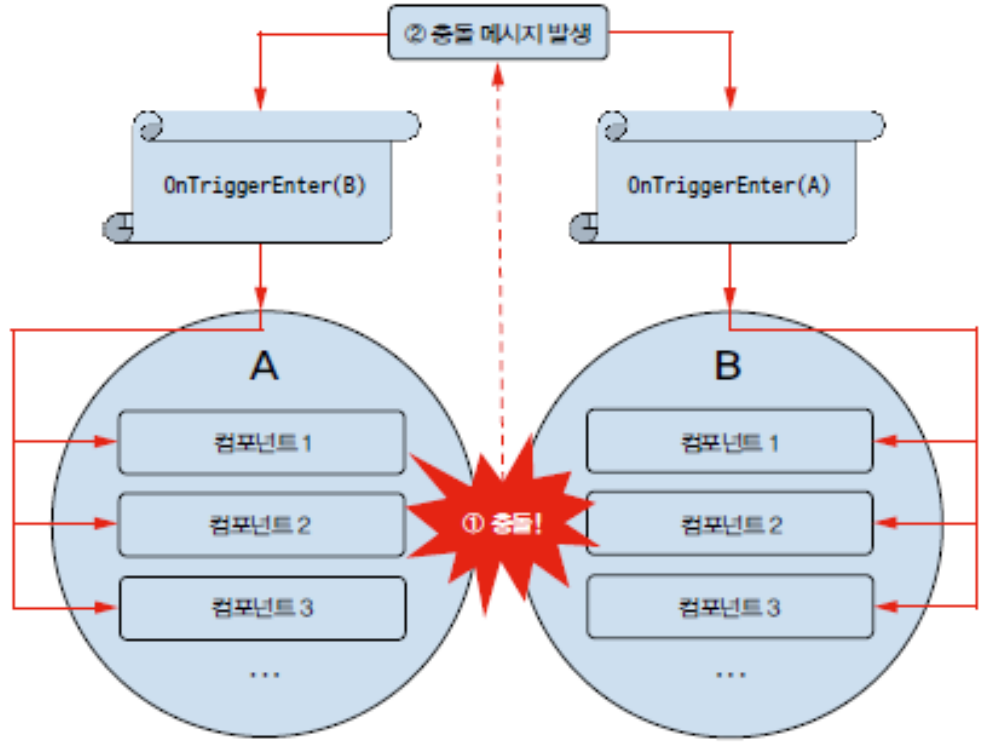
```
void Destroy(Object obj);
```

```
void Destroy(Object obj, float t);
```

7.3 탄알의 충돌 처리

◦ 7.3.1 충돌 이벤트 메서드

- 유니티에서 게임 오브젝트 간의 충돌을 어떻게 감지하는지 알아봄.
- 충돌 메시지를 통해 게임 오브젝트와 해당 게임 오브젝트에 추가된 컴포넌트들은 충돌 사실을 알게 되고 충돌에 대응하는 메서드를 실행함.



▶ 충돌한 게임 오브젝트에 전파되는 충돌 메시지

OnCollision 계열 : 일반 충돌

일반적인 콜라이더를 가진 두 게임 오브젝트가 충돌할 때 자동으로 실행됩니다. 충돌한 두 콜라이더는 서로 통과하지 않고 밀어냅니다.

- `OnCollisionEnter(Collision collision)` : 충돌한 순간
- `OnCollisionStay(Collision collision)` : 충돌하는 동안
- `OnCollisionExit(Collision collision)` : 충돌했다가 분리되는 순간

OnTrigger 계열 : 트리거 충돌

충돌한 두 게임 오브젝트의 콜라이더 중 최소 하나가 트리거 콜라이더라면 자동으로 실행됩니다. 이 경우 두 게임 오브젝트가 충돌했을 때 서로 그대로 통과합니다.

- `OnTriggerEnter(Collider other)` : 충돌한 순간
- `OnTriggerStay(Collider other)` : 충돌하는 동안
- `OnTriggerExit(Collider other)` : 충돌했다가 분리되는 순간

7.3 탄알의 충돌 처리

- 7.3.2 탄알에 충돌 감지 구현
  - Bullet 스크립트에 충돌 이벤트 메서드로 OnTriggerEnter( )를 작성해야 함.

[과정 01] OnTriggerEnter( ) 메서드 작성

① Bullet 스크립트에 OnTriggerEnter( ) 메서드를 추가하고 다음과 같이 완성

```
// using 문 생략

public class Bullet : MonoBehaviour {
    public float speed = 8f; // 탄알 이동 속력
    private Rigidbody bulletRigidbody; // 이동에 사용할 리지드바디 컴포넌트

    void Start() {

   // Start() 메서드 내용 생략
}

// 트리거 충돌 시 자동으로 실행되는 메서드
void OnTriggerEnter(Collider other) {
    // 충돌한 상대방 게임 오브젝트가 Player 태그를 가진 경우
    if (other.tag == "Player")
    {
        // 상대방 게임 오브젝트에서 PlayerController 컴포넌트 가져오기
        PlayerController playerController = other.GetComponent<PlayerController>();

        // 상대방으로부터 PlayerController 컴포넌트를 가져오는 데 성공했다면
        if (playerController != null)
        {
            // 상대방 PlayerController 컴포넌트의 Die() 메서드 실행
            playerController.Die();
        }
    }
  }
}
```

7.3 탄알의 충돌 처리

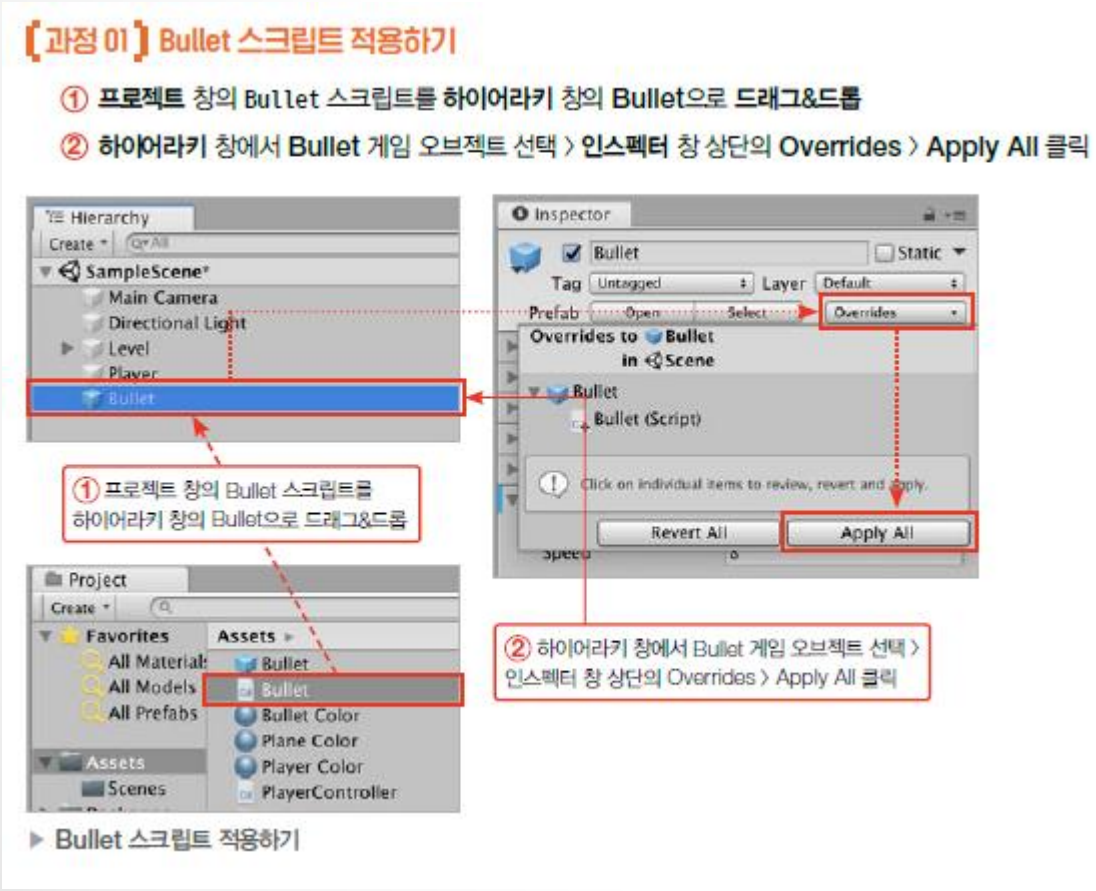
- 7.3.3 완성된 전체 Bullet 스크립트
  - 지금까지 완성된 전체 Bullet 스크립트는 다음과 같음.

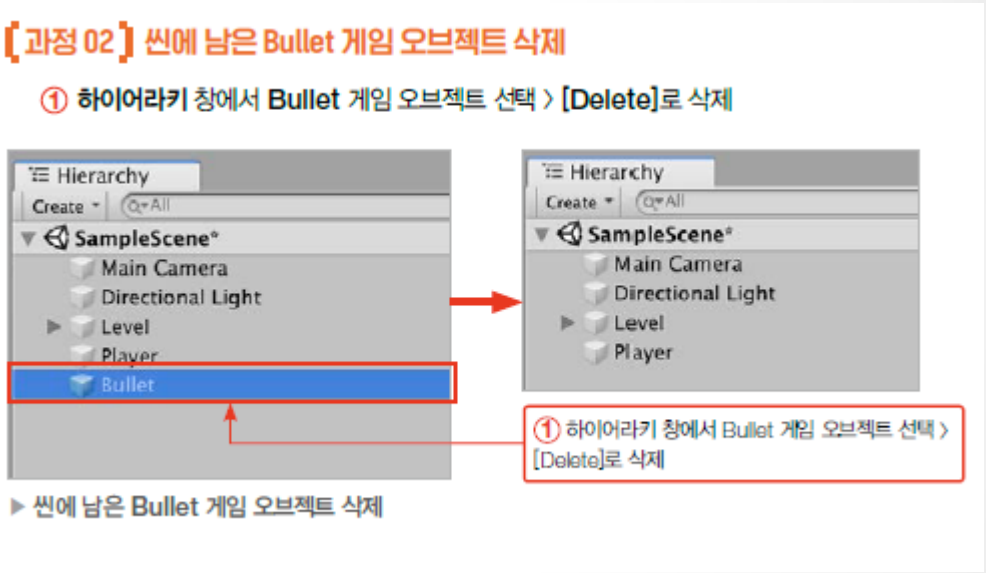
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Bullet : MonoBehaviour {
    public float speed = 8f; // 탄알 이동 속력
    private Rigidbody bulletRigidbody; // 이동에 사용할 리지드바디 컴포넌트

    void Start() {
        // 게임 오브젝트에서 Rigidbody 컴포넌트를 찾아 bulletRigidbody에 할당
        bulletRigidbody = GetComponent<Rigidbody>();
        // 리지드바디의 속도 = 앞쪽 방향 * 이동 속력
        bulletRigidbody.velocity = transform.forward * speed;

        // 3초 뒤에 자신의 게임 오브젝트 파괴
        Destroy(gameObject, 3f);
    }

    // 트리거 충돌 시 자동으로 실행되는 메서드
    void OnTriggerEnter(Collider other) {
        // 충돌한 상대방 게임 오브젝트가 Player 태그를 가진 경우
        if (other.tag == "Player")
        {
            // 상대방 게임 오브젝트에서 PlayerController 컴포넌트 가져오기
            PlayerController playerController = other.GetComponent<PlayerController>();

            // 상대방으로부터 PlayerController 컴포넌트를 가져오는 데 성공했다면
            if (playerController != null)
            {
                // 상대방 PlayerController 컴포넌트의 Die() 메서드 실행
                playerController.Die();
            }
        }
    }
}
```

7.3 탄알의 충돌 처리

- 7.3.4 Bullet 게임 오브젝트 완성

  - 완성된 Bullet 스크립트를 Bullet 게임 오브젝트에 컴포넌트로 추가함.

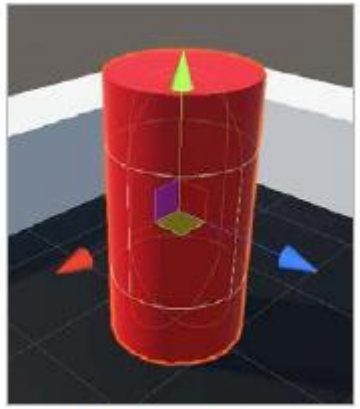




7.4 탄알 생성기 준비

- 7.4.1 Bullet Spawner 게임 오브젝트 준비

  - 탄알 생성기가 될 게임 오브젝트를 만들고 배치함. 탄알 생성기를 만드는 데 원기둥 게임 오브젝트를 활용하겠음.



▶ 탄알 생성기의 모습

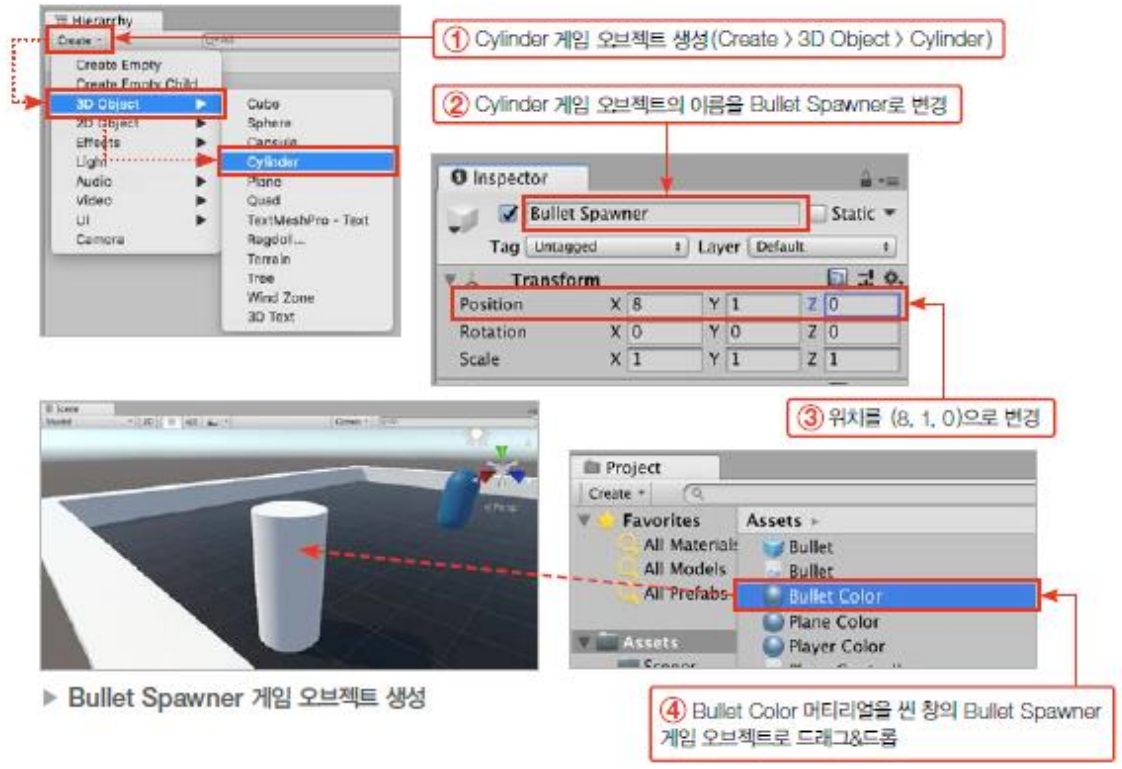
[과정 01] Bullet Spawner 게임 오브젝트 생성

① Cylinder 게임 오브젝트 생성(하이어라키 창에서 Create > 3D Object > Cylinder)

② 생성된 Cylinder 게임 오브젝트의 이름을 Bullet Spawner로 변경

③ Bullet Spawner 게임 오브젝트의 위치를 (8, 1, 0)으로 변경

④ 프로젝트 창의 Bullet Color 머티리얼을 씬 창의 Bullet Spawner 게임 오브젝트로 드래그&드롭



▶ Bullet Spawner 게임 오브젝트 생성

7.5 탄알 생성기 스크립트 만들기

- 7.5.1 BulletSpawner의 변수

  - 탄알 생성기 Bullet Spawner 게임 오브젝트가 탄알을 생성할 수 있게 하는 BulletSpawner 스크립트를 만듬.
  - BulletSpawner 스크립트에는 다음 역할을 수행할 변수들이 필요함. BulletSpawner 스크립트에 선언합니다.

[과정 01] BulletSpawner 스크립트 생성

① **프로젝트** 창에서 **Create** > **C# Script** 클릭
② 생성된 스크립트 **이름**을 BulletSpawner으로 변경
③ BulletSpawner 스크립트를 **더블 클릭**으로 **열기**

- 생성할 탄알의 원본
- 탄알을 발사하여 맞출 대상
- 탄알을 생성하는 시간 간격

[과정 01] BulletSpawner의 변수 선언하기

① BulletSpawner 스크립트를 다음과 같이 수정

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

public class BulletSpawner : MonoBehaviour {
    public GameObject bulletPrefab; // 생성할 탄알의 원본 프리팹
    public float spawnRateMin = 0.5f; // 최소 생성 주기
    public float spawnRateMax = 3f; // 최대 생성 주기

    private Transform target; // 발사할 대상
    private float spawnRate; // 생성 주기
    private float timeAfterSpawn; // 최근 생성 시점에서 지난 시간

    void Start() {

    }

    void Update() {

    }
}
```

7.5 탄알 생성기 스크립트 만들기

- 변수들의 역할은 다음과 같음.

public 변수

- bulletPrefab : 탄알을 생성하는 데 사용할 원본 프리팹
- spawnRateMin : 새 탄알을 생성하는 데 걸리는 시간의 최솟값
- spawnRateMax : 새 탄알을 생성하는 데 걸리는 시간의 최댓값

private 변수

- target : 조준할 대상 게임 오브젝트의 트랜스폼 컴포넌트
- spawnRate : 다음 탄알을 생성할 때까지 기다릴 시간. spawnRateMin과 spawnRateMax 사이의 랜덤 값으로 설정됨.
- timeAfterSpawn : 마지막 탄알 생성 시점부터 흐른 시간을 표시하는 '타이머'

7.5 탄알 생성기 스크립트 만들기

- 7.5.2 Start( ) 메서드
  - Start( ) 메서드를 완성함.

[과정 01] BulletSpawner 스크립트의 Start ( ) 메서드 완성하기

① BulletSpawner 스크립트의 Start ( ) 메서드를 다음과 같이 완성

```
void Start() {
    // 최근 생성 이후의 누적 시간을 0으로 초기화
    timeAfterSpawn = 0f;
    // 탄알 생성 간격을 spawnRateMin과 spawnRateMax 사이에서 랜덤 지정
    spawnRate = Random.Range(spawnRateMin, spawnRateMax);
    // PlayerController 컴포넌트를 가진 게임 오브젝트를 찾아 조준 대상으로 설정
    target = FindObjectOfType<PlayerController>().transform;
}
```

FindObjectOfType( ) 메서드

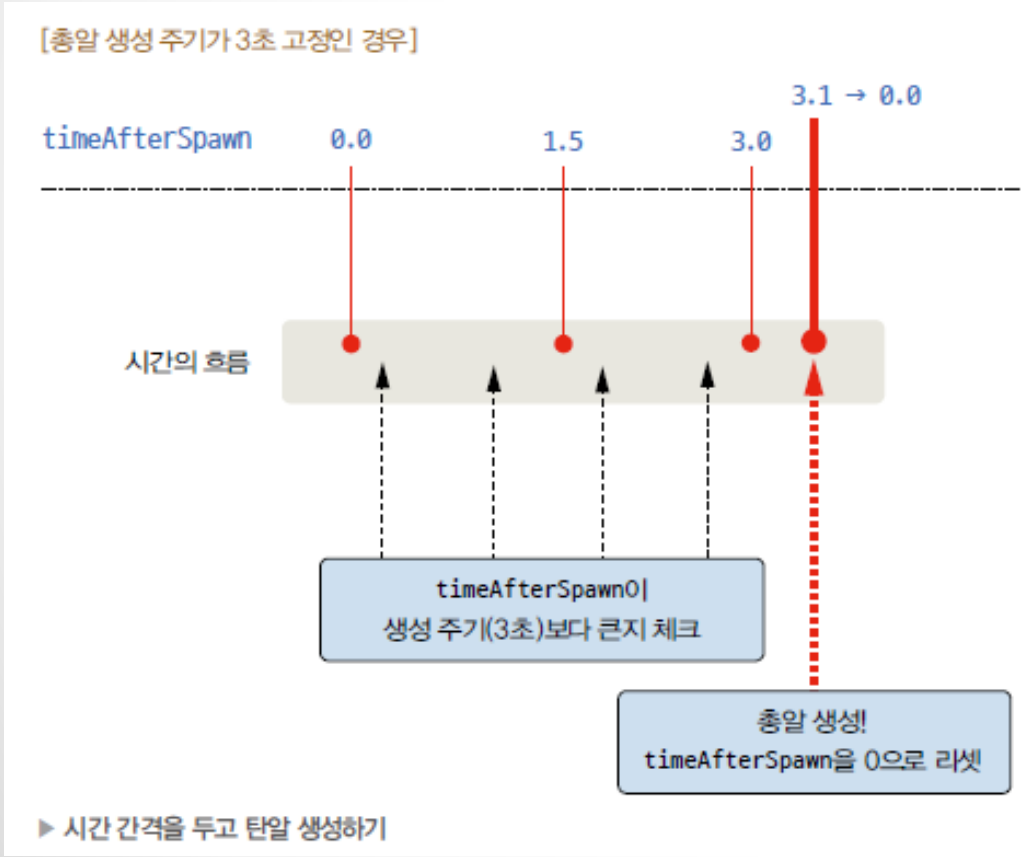
target에는 탄알이 날아갈 대상, 즉 Player 게임 오브젝트의 트랜스폼 컴포넌트가 할당되어야 합니다. Player 게임 오브젝트의 트랜스폼으로 Player 게임 오브젝트의 위치를 파악할 수 있기 때문입니다.

```
target = FindObjectOfType<PlayerController>().transform;
```

```
PlayerController playerController = FindObjectOfType<PlayerController>();
target = playerController.transform;
```

7.5 탄알 생성기 스크립트 만들기

- 7.5.3 일정 주기로 실행 반복하기
  - Update( ) 메서드에서 탄알을 생성할 것임.
  - Update( ) 메서드에 탄알 생성 코드를 넣으면 탄알이 1초에 수십 개씩 쉴 새 없이 생성됨.
  - 탄알을 생성하기 전에 마지막으로 탄알을 생성한 시점에서 누적된 시간을 저장하는 변수 timeSpawnRate를 체크함.



Update()의 실행 시간 간격

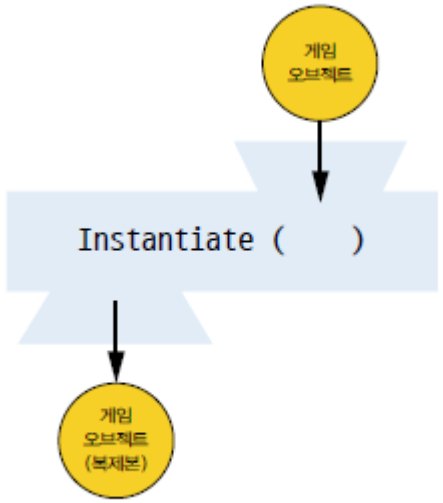
Update()는 화면이 한 번 갱신될 때마다 한 번 실행됩니다. 따라서 마지막 Update()가
된 시점과 현재 Update()가 실행된 시점 사이의 시간 간격이 프레임이 새로 그려지는 데
는 시간입니다.

7.5 탄알 생성기 스크립트 만들기

- 7.5.4 Time.deltaTime
  - 초당 프레임은 컴퓨터 성능에 따라 다름.
  - Update ( ) 실행 사이의 시간 간격을 알기 위해 내장 변수 Time.deltaTime을 사용함.
  - Time.deltaTime에는 이전 프레임과 현재 프레임 사이의 시간 간격이 자동으로 할당됨.
  - Update( ) 메서드에서 어떤 변수에 Time.deltaTime 값을 계속 누적하면 특정 시점으로부터 시간이 얼마나 흘렀는지 표현할 수 있음.

7.5 탄알 생성기 스크립트 만들기

- 7.5.5 Instantiate( ) 메서드
  - 탄알을 복제 생성하는 데 Instantiate( ) 메서드를 사용할 것임.
  - 유니티는 게임 도중에 실시간으로 오브젝트를 생성할 때 Instantiate( ) 메서드를 사용함.



▶ Instantiate( ) 메서드의 동작

```
Instantiate(bulletPrefab);
```

```
Instantaite(원본, 위치, 회전);
```

- transform.position : 자신의 위치
- transform.rotation : 자신의 회전

```
Instantiate(bulletPrefab, transform.position, transform.rotation);
```

7.5 탄알 생성기 스크립트 만들기

- 7.5.6 Update( ) 메서드

  - Update( ) 메서드에서 다음과 같은 처리를 하여 주기적으로 탄알을 생성하는 처리를 구현함.

[과정 01] BulletSpawner 스크립트의 Update ( ) 메서드 완성하기

① BulletSpawner 스크립트의 Update ( ) 메서드를 다음과 같이 완성

```
void Update() {
    // timeAfterSpawn 갱신
    timeAfterSpawn += Time.deltaTime;

    // 최근 생성 시점에서부터 누적된 시간이 생성 주기보다 크거나 같다면
    if (timeAfterSpawn >= spawnRate)
    {
        // 누적된 시간을 리셋
        timeAfterSpawn = 0f;

        // bulletPrefab의 복제본을
        // transform.position 위치와 transform.rotation 회전으로 생성
        GameObject bullet
            = Instantiate(bulletPrefab, transform.position, transform.rotation);
        // 생성된 bullet 게임 오브젝트의 정면 방향이 target을 향하도록 회전
        bullet.transform.LookAt(target);

        // 다음번 생성 간격을 spawnRateMin, spawnRateMax 사이에서 랜덤 지정
        spawnRate = Random.Range(spawnRateMin, spawnRateMax);
    }
}
```

탄알 복제 생성하기

```
if (timeAfterSpawn >= spawnRate)
{
    timeAfterSpawn = 0f;

    GameObject bullet
        = Instantiate(bulletPrefab, transform.position, transform.rotation);
    bullet.transform.LookAt(target);

    spawnRate = Random.Range(spawnRateMin, spawnRateMax);
}
```

다음 생성 시점 변경하기

```
spawnRate = Random.Range(spawnRateMin, spawnRateMax);
```

7.5 탄

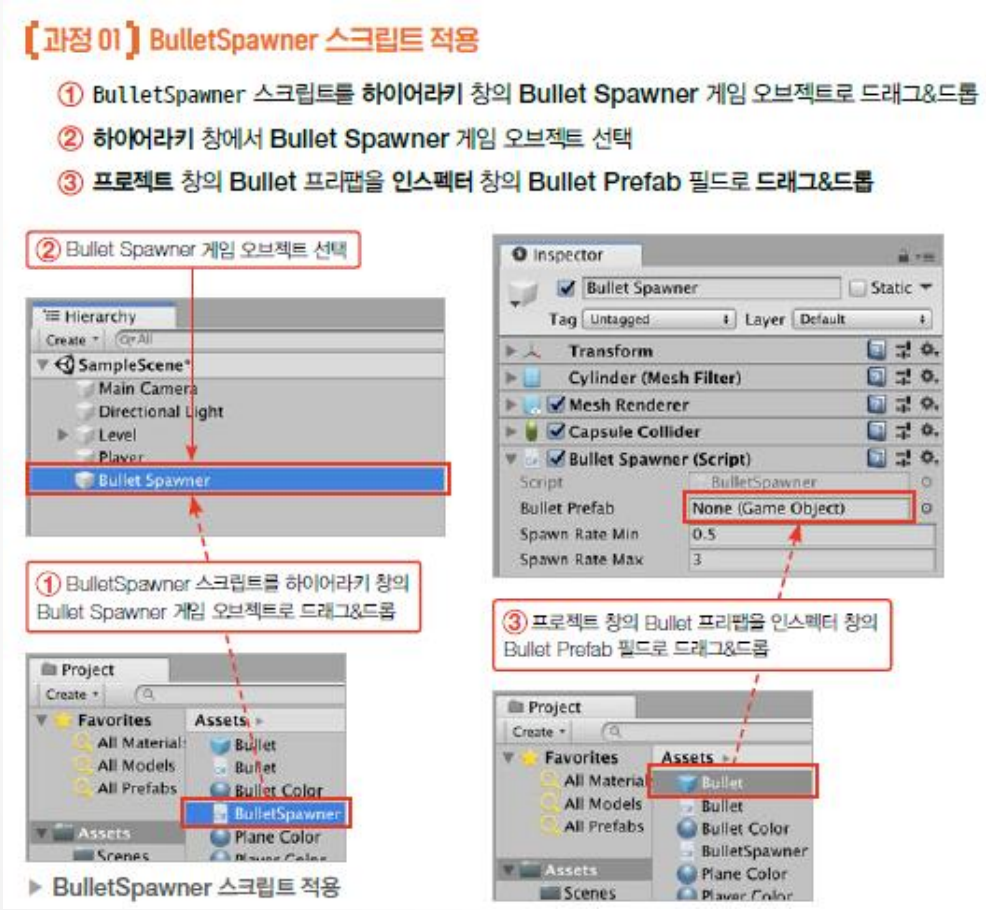
- 7.5.7 완성된 탄알 생성기 스크립트

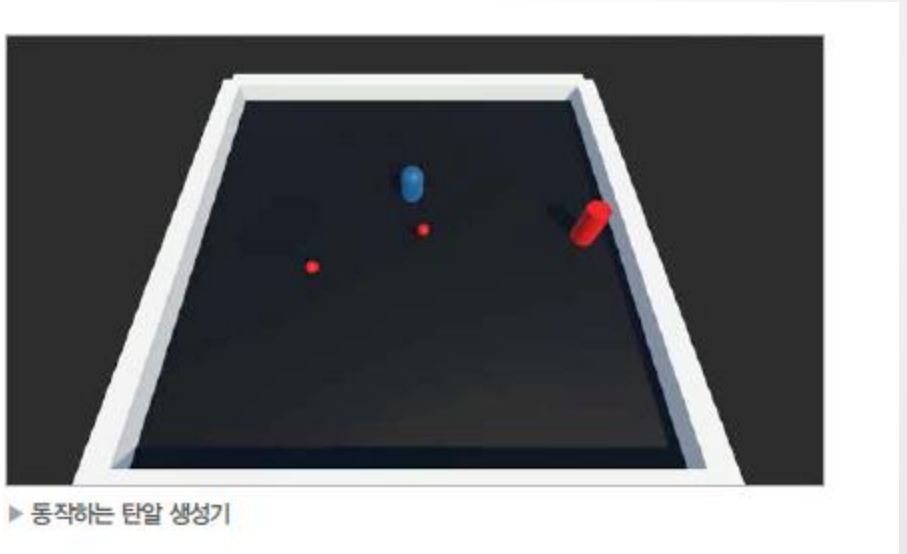
  - 지금까지 완성한 전체 BulletSpawner 스크립트는 다음과 같음.

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

public class BulletSpawner : MonoBehaviour {
    public GameObject bulletPrefab; // 생성할 탄알의 원본 프리팹
    public float spawnRateMin = 0.5f; // 최소 생성 주기
    public float spawnRateMax = 3f; // 최대 생성 주기

    private Transform target; // 발사할 대상
    private float spawnRate; // 생성 주기
    private float timeAfterSpawn; // 최근 생성 시점에서 지난 시간

    void Start() {
        // 최근 생성 이후의 누적 시간을 0으로 초기화
        timeAfterSpawn = 0f;
        // 탄알 생성 간격을 spawnRateMin과 spawnRateMax 사이에서 랜덤 지정
        spawnRate = Random.Range(spawnRateMin, spawnRateMax);
        // PlayerController 컴포넌트를 가진 게임 오브젝트를 찾아 조준 대상으로 설정
        target = FindObjectOfType<PlayerController>().transform;
    }

    void Update() {
        // timeAfterSpawn 갱신
        timeAfterSpawn += Time.deltaTime;

        // 최근 생성 시점에서부터 누적된 시간이 생성 주기보다 크거나 같다면
        if (timeAfterSpawn >= spawnRate)
        {
            // 누적된 시간을 리셋
            timeAfterSpawn = 0f;

            // bulletPrefab의 복제본을
            // transform.position 위치와 transform.rotation 회전으로 생성
            GameObject bullet
                = Instantiate(bulletPrefab, transform.position, transform.rotation);
            // 생성된 bullet 게임 오브젝트의 정면 방향이 target을 향하도록 회전
            bullet.transform.LookAt(target);

            // 다음번 생성 간격을 spawnRateMin, spawnRateMax 사이에서 랜덤 지정
            spawnRate = Random.Range(spawnRateMin, spawnRateMax);

        }
    }
}
```

7.5 탄알 생성기 스크립트 만들기

- 7.5.8 BulletSpawner 스크립트 적용하기
  - 완성된 BulletSpawner 스크립트를 Bullet Spawner 게임 오브젝트에 컴포넌트로 추가함.
  - BulletSpawner 컴포넌트가 동작할 수 있도록 설정함.



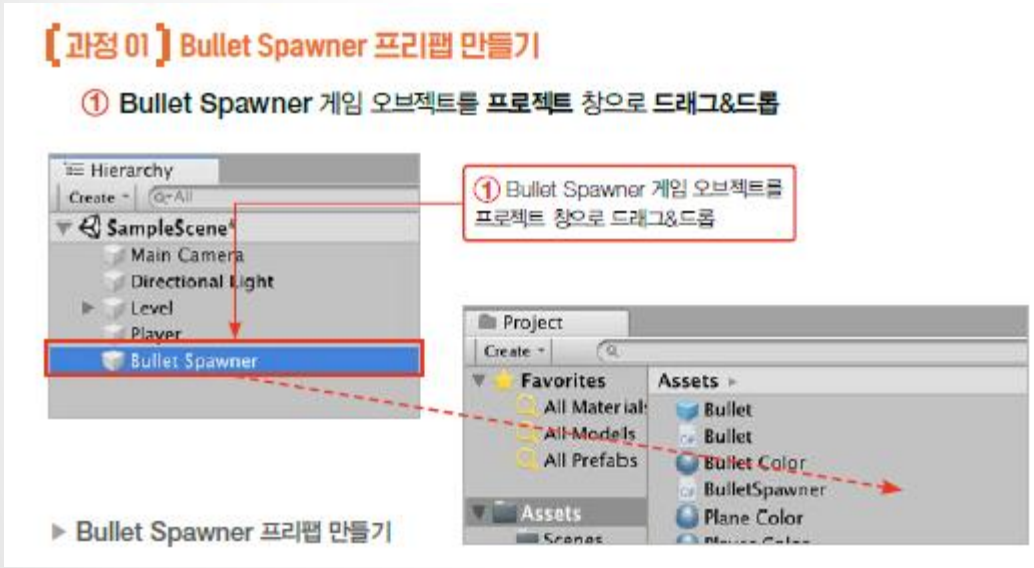


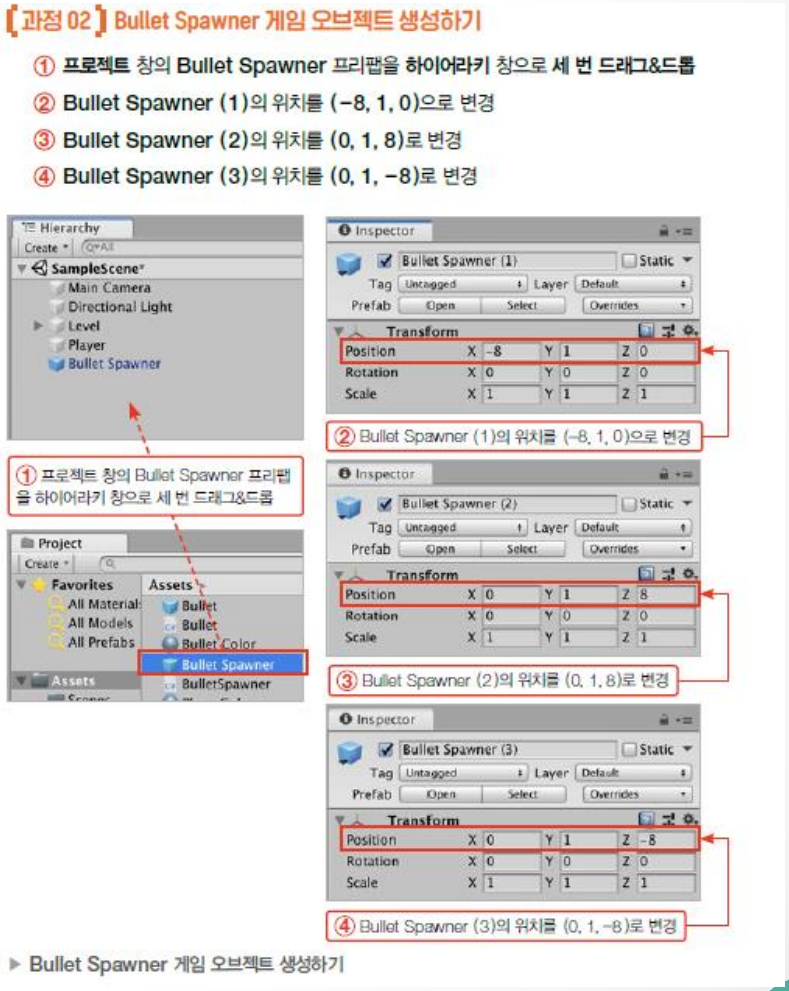
▶ 동작하는 탄알 생성기

7.5 탄알 생성기 스크립트 만들기

- 7.5.9 탄알 생성기 배치하기

  - 완성된 탄알 생성기를 프리팹으로 만들고, 씬에 탄알 생성기를 여러 개 배치함.

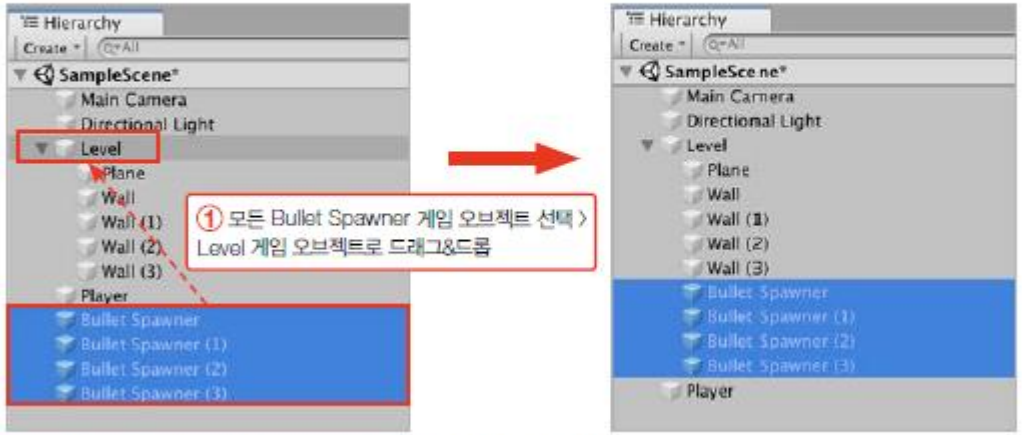




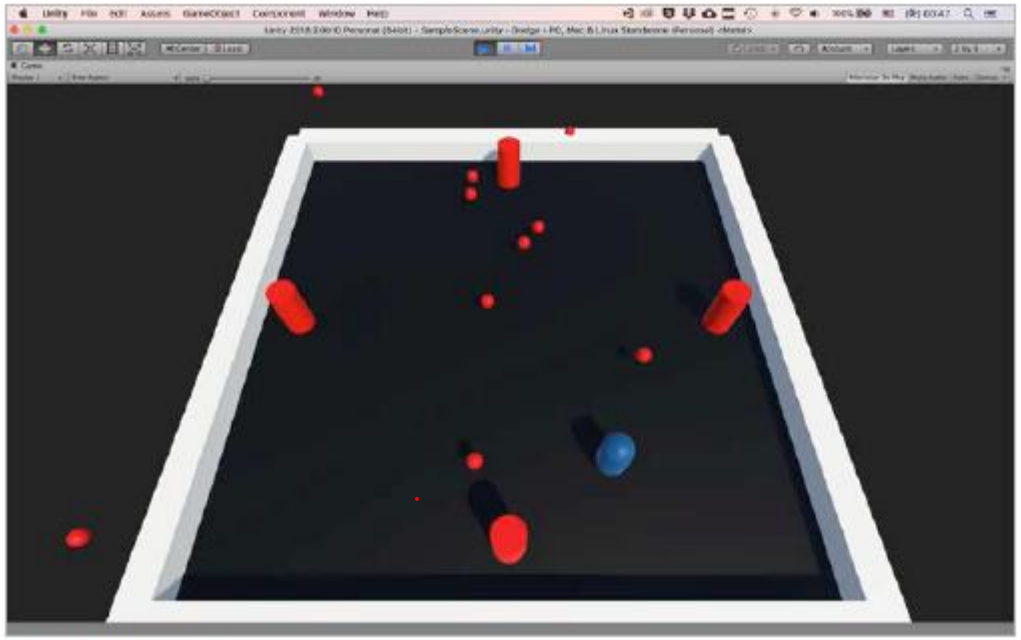
7.5 탄알 생성기 스크립트 만들기

[과정 03] Bullet Spawner 게임 오브젝트를 Level의 자식으로 넣기

① **하이어라키** 창에서 모든 **Bullet Spawner** 게임 오브젝트 선택 > **Level** 게임 오브젝트로 **드래그&드롭**



▶ Bullet Spawner 게임 오브젝트를 Level의 자식으로 넣기



▶ 사방에서 날아오는 탄알