

# CIEN Unity 3D

중급반 (화요일 오후 7시)

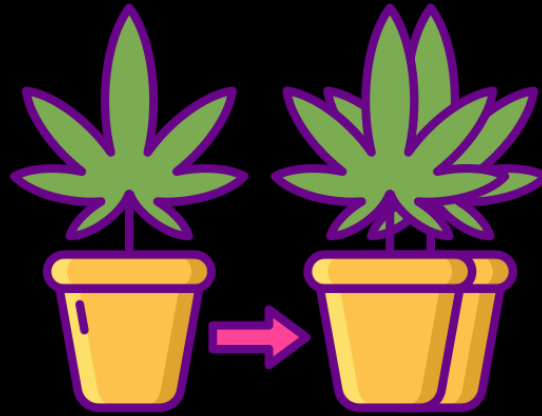
고주형



# 목차

1. 프리팹(Prefab)
  - 미사일 프리팹 만들기
  - 미사일 생성하기
    - Instantiate 함수
2. 게임오브젝트 찾기
  - Find - name, tag, type
3. 코루틴을 위해 알아야할 C# 문법
  - 람다식, IEnumerator, yield
4. 코루틴(Coroutine)이란?
5. 미사일 쏘는 적기 만들기
  - 미사일에 맞으면 "아야" 출력

실습. <https://github.com/CIEN-Club/workshop-guestbook>의 README.md를 읽고 저장소에 Push해서 방명록  
담기기



# 1. 프리팹 (Prefab)

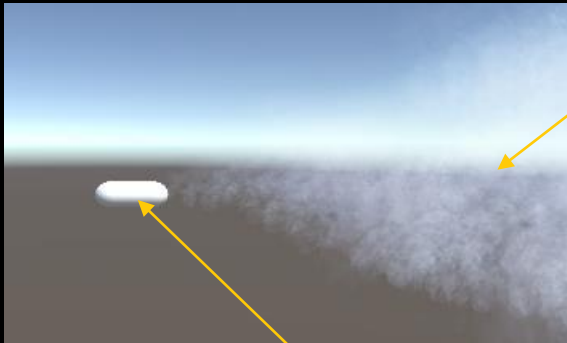
편하게 재사용하자!

# 복습 프리팸 (Prefab)

- 내가 조작해 놓은 게임오브젝트의 구성된 기능들을 하나로 묶은 템플릿.
- 템플릿과 같은 용도 많이 쓰임.
  - 유니티 월드상에 배치된 프리팸들은 기본적으로 원본 데이터를 기준으로 동작한다
  - 하지만 고유한 값은 따로 존재한다

# 미사일을 만들어보자

- 자신만의 미사일을 만들자!



## White Smoke Particle System



Xenomash Games

★★★★★ 5 | 67 Reviews

FREE

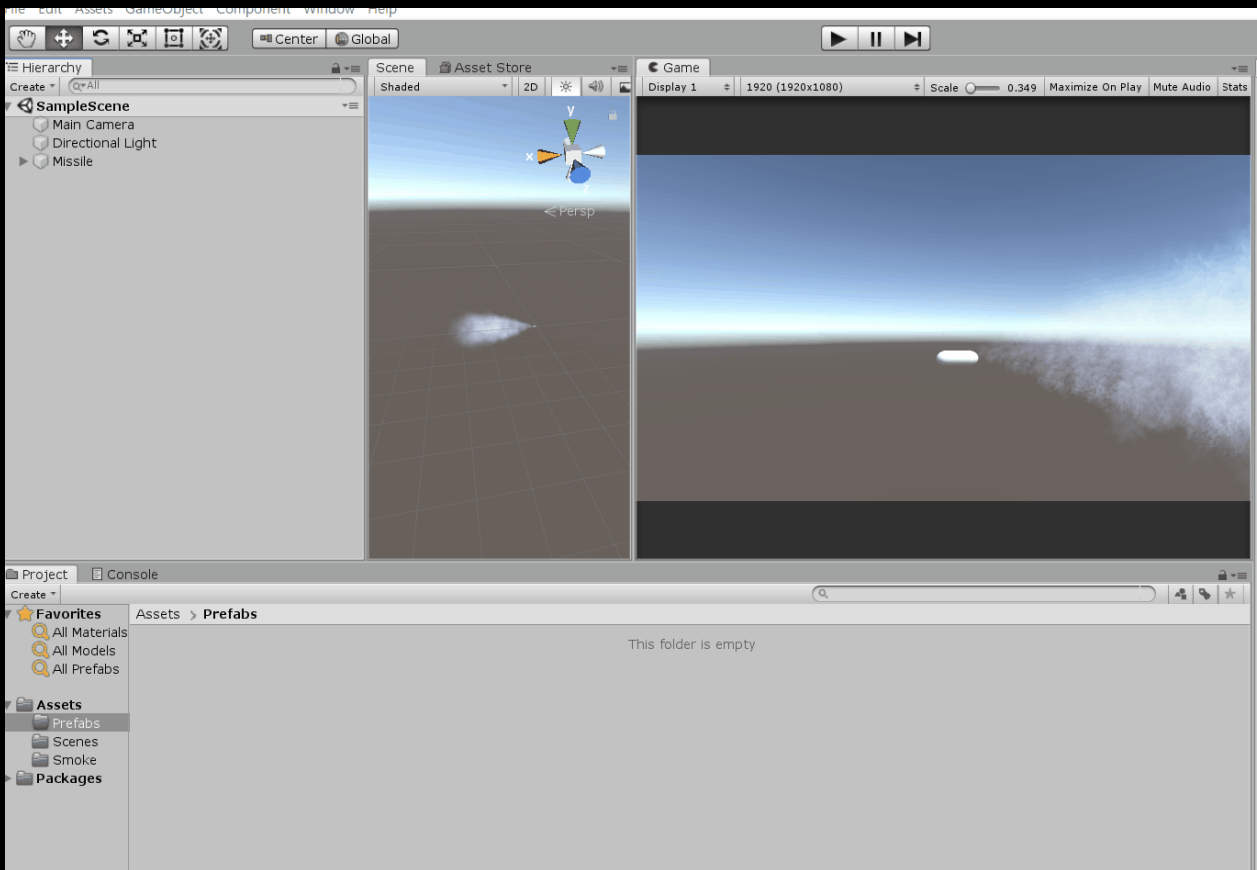
Import

```
using UnityEngine;

public class MissileCtrl : MonoBehaviour
{
    // Start is called before the first frame update
    void Start()
    {
        GetComponent<Rigidbody>().velocity = -transform.right;
    }
}
```

# 매번 다시 만들어야 할까?

- LL, 프리팹으로 만들자.



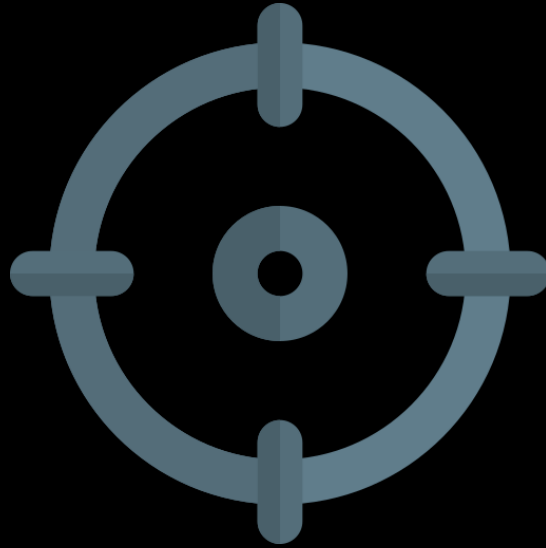
# 미사일을 생성해보자

- **Instantiate** 함수
  - 원본 게임오브젝트를 복제하는 함수
  - 어떻게 사용할지 모르니 **Object**형으로 반환

```
public GameObject player;
public GameObject originalPrefab;
GameObject temp;

void Update()
{
    if(Input.GetButtonDown("Fire1"))
    {
        temp = Instantiate(originalPrefab) as GameObject;
        temp.name = "Missile";
        temp.transform.position = player.transform.position;
    }
}
```





## 2. 게임오브젝트 찾기

특정 게임오브젝트를 추적하려면?



# 플레이어를 공격하도록 하려면?

- 플레이어의 게임오브젝트를 찾아야 한다
- 특정 게임오브젝트는 어떻게 찾을 수 있을까?

# 게임오브젝트 찾는 방법

어떤 것을 써야 할까?

Public 변수에 Drag&Drop

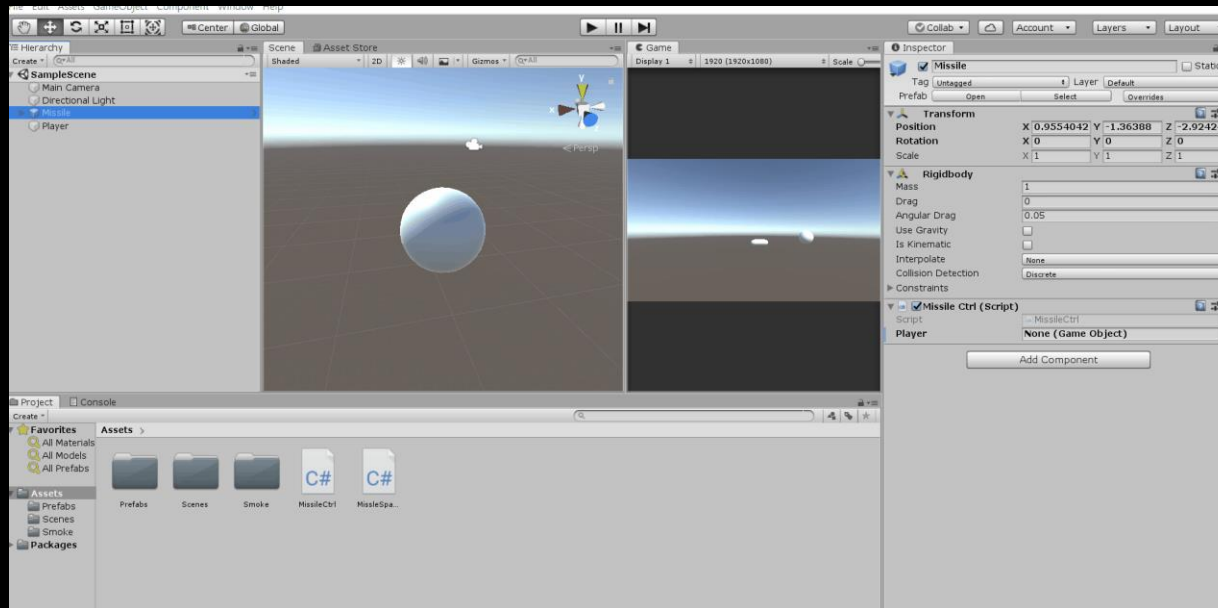
게임오브젝트 이름으로 찾기

게임오브젝트 Tag명으로 찾기

Type으로 찾기

# 방법 1

## Public 변수에 Drag&Drop



- 제일 빠르다.
- 단점
  - 새로 생성하는 게임오브젝트에는 D&D할 수가 없음
  - 매번 설정해주기 귀찮음

## 방법 2

# 게임오브젝트 이름으로 찾기

```
GameObject.Find("Player");
```

- 단점
  - 게임오브젝트의 이름을 바꾸면 못 찾음.
  - 게임오브젝트의 이름이 고정 됨.
  - 나중에 가면 어떤 오브젝트가 이름으로 찾고 있는지 기억나지가 않아서 모든 게임 오브젝트의 이름 바꾸기가 어려워 짐.
- 주의: string concatenation

```
go.name = "MyObject" + count;
```

## 방법 3

# 게임오브젝트 Tag명으로 찾기

```
GameObject.FindGameObjectWithTag("Player")
```

- 단점
  - Code상에 잘 보이지 않음.
  - 게임오브젝트에 Tag가 달렸는지 잘 노출되지 않는다.
  - 협업할 때 다른 사람이 이해하기 어려움.
  - 새로운 게임오브젝트를 추가할 때 Tag를 넣는 것을 깜박하기 쉬움.

## 방법 4

# Type으로 찾기

```
GameObject.FindObjectOfType<MissileCtrl>()
```

- 단점
  - 위의 방법 중 제일 느리다

# Find는 느리다

- 나중에 가면 GameObject가 엄청 늘어나는데 그 중에서 하나를 찾으려면? 느리다!!

계속 찾지 말고

미리 저장해두자!

한 프레임 정도는 상관 X

- Start()에서 Find() 미리 하기
- Instantiate를 할 때 배열에 넣어두기
- (Drag and Drop)
- ...

# 미리 찾기

- Cache it!
- Start에서 미리 찾아주자. (Update에서 사용 X)

```
using UnityEngine;

public class MissileCtrl : MonoBehaviour
{
    private GameObject player;

    // Start is called before the first frame update
    void Start()
    {
        player = GameObject.Find("Player");
    }

    // use cached player
    // ...
}
```



# 나중에 사용할 것은 저장해두자

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

public class MissileSpawner : MonoBehaviour
{
    public GameObject originalPrefab;
    GameObject temp;
    List<GameObject> useLater = new List<GameObject>();

    // Update is called once per frame
    void Spawn()
    {
        temp = Instantiate(originalPrefab);
        useLater.Add(temp);
    }
}
```

```
public int hp;
void Start()
{
    hp = 100;
    StartCoroutine(DeadOrAlive());
}

IEnumerator DeadOrAlive()
{
    Debug.Log("Waiting for player to be dead");
    yield return new WaitForSeconds(10);
    Debug.Log("Player is dead");
}
```

## 3. C# 문법

코루틴을 위해 알아야할 C# 문법들

Lambda IEnumerator

# 람다식

- (인자) => 식
- (인자) => { 식들 }
- 반환 0
  - Func<T, TResult>로 변환 가능
- 반환 X (void)
  - Action<T1, T2>로 변환 가능
- 참고: <https://docs.microsoft.com/ko-kr/dotnet/csharp/programming-guide/statements-expressions-operators/lambda-expressions>

# 간단한 예시

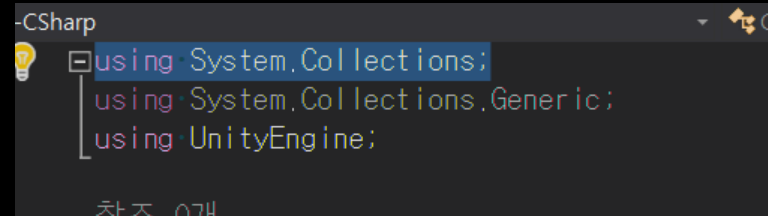
- "Hello, World!"를 출력하는 람다식
  - `() => Debug.Log("Hello, World!");`
- 더하기 람다식
  - `(x, y) => x + y;`

```
public int hp;
void Start()
{
    hp = 100;
    StartCoroutine(DeadOrAlive());
}

IEnumerator DeadOrAlive()
{
    Debug.Log("Waiting for player to be dead");
    yield return new WaitForSeconds(10);
    Debug.Log("Player is dead");
}
```

# 열거자 (IEnumerator)

- System.Collection에 있다
- 열거하기 위해 사용한다
- 열거자를 구현하면
  - MoveNext()
  - Current



A screenshot of a C# code editor window titled "-CSharp". The code contains three using statements: `using System.Collections;`, `using System.Collections.Generic;`, and `using UnityEngine;`. The first line is highlighted with a blue selection bar. A lightbulb icon is visible on the left side of the editor, and a small icon is in the top right corner. At the bottom, there is a status bar with the text "참조 0개".

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

참조 0개

# 사용해보자

```
IEnumerator GiveMeNumber()
{
    Debug.Log("Start IEnumerator");
    yield return 1;
    Debug.Log("Start IEnumerator again");
    yield return 2;
    Debug.Log("Start IEnumerator again");
    yield return 3;
    Debug.Log("End IEnumerator");
}

void Start()
{
    IEnumerator enumerator = GiveMeNumber();
    for(int i=0; i<4; i++)
    {
        Debug.Log(i + "번째 Current: " + enumerator.Current);
        Debug.Log(i + "번째 MoveNext의 반환: " + enumerator.MoveNext());
    }
}
```

# 결과

[18:12:15] 0번째 Current: <b>Null</b> UnityEngine.Debug:Log(Object)	1
[18:12:15] Start IEnumerator UnityEngine.Debug:Log(Object)	1
[18:12:15] 0번째 MoveNext의 반환: True UnityEngine.Debug:Log(Object)	1
[18:12:15] 1번째 Current: 1 UnityEngine.Debug:Log(Object)	1
[18:12:15] Start IEnumerator again UnityEngine.Debug:Log(Object)	1
[18:12:15] 1번째 MoveNext의 반환: True UnityEngine.Debug:Log(Object)	1
[18:12:15] 2번째 Current: 2 UnityEngine.Debug:Log(Object)	1
[18:12:15] Start IEnumerator again UnityEngine.Debug:Log(Object)	1
[18:12:15] 2번째 MoveNext의 반환: True UnityEngine.Debug:Log(Object)	1
[18:12:15] 3번째 Current: 3 UnityEngine.Debug:Log(Object)	1
[18:12:15] End IEnumerator UnityEngine.Debug:Log(Object)	1
[18:12:15] 3번째 MoveNext의 반환: False UnityEngine.Debug:Log(Object)	1

StartCoroutine

## 4. 코루틴

코루틴이 뭘까요?



# 코루틴이란?

- Entry Point가 여러 개인 함수
- `yield`에서 반환되고 다시 이어서 시작할 수 있음
- 협력하는 함수

# 유니티에서 코루틴이란?

- IEnumerator Interface를 반환하는 함수
- 유니티는 Single Thread
  - Thread 대신 사용
- 유니티랑 협동한다

# A. 스크립트/유니티 엔진

`StartCoroutine(IEnumerator)`

- 엔진이 `MoveNext`를 바로 호출한다

## B. 코루틴

코루틴이 실행된다

- `yield return`에서 값을 반환한다
- 대기한다

# A. 유니티 엔진

- 반환 값을 받고 수납한다
- 때가 됐을 때 다시 `MoveNext()` 호출

## B. 코루틴

코루틴이 실행된다

- `yield return`에서 값을 반환한다
- 대기한다

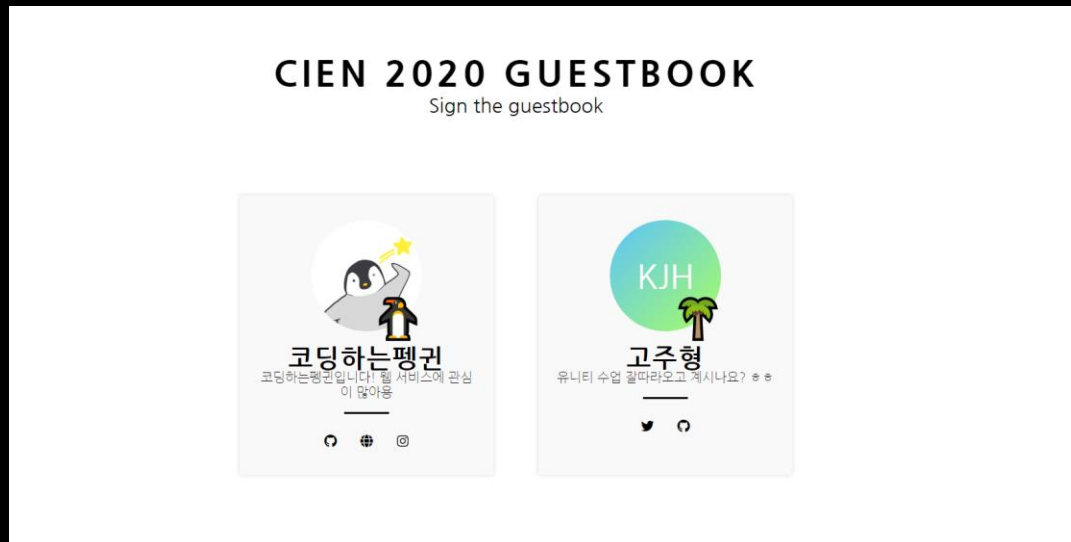
# ... 더 이상 `return`할 것이 없으면

- 코루틴 종료

# 다른 사람의 저장소에 Push해서 방명록을 작성해보자!

Git Branch 학습용 숙제입니다

- <https://github.com/CIEN-Club/workshop-guestbook>의 README.md를 읽고 저장소에 Push해서 방명록을 남기자.
- 방명록을 Commit하면 자동으로 웹 사이트에 등록되도록 프로그래밍되어 있다.





# 참고 문서

- Icons made by [Flat Icons](#), [Pixel perfect](#) from [www.flaticon.com](http://www.flaticon.com)