



CIEN

"Unity 3D"

Unity 2018.4.9f1 (64-bit) 고주형

<https://github.com/lamGrooooooot/cien-unity-2019>

2019/11/06

Week 1

- 유니티 소개
- 유니티 설치
- 화면 구성
- 조작 방법
- 간단한 실습
- 씬(Scene)
- 컴포넌트(Component)
- 게임 오브젝트(GameObject)
- 돌 굴리기
- 스크립트(Script)



유니티 소개

모두를 위한 Unity

여러분의 비전을 지금 바로 현실로 만드세요. Unity의 실시간 3D 플랫폼은 제작, 운영, 수익 창출에 필요한 모든 것을 갖추고 있습니다.

[시작하기](#)[자세히 알아보기](#)[게임](#)[자동차, 운송 및 제조](#)[영화, 애니메이션 및
시네마틱](#)[설계, 엔지니어링 및
건축](#)

유니티로 할 수 있는 것들

- Android App
- iOS App
- PC
- Web
- VR / AR
- ...

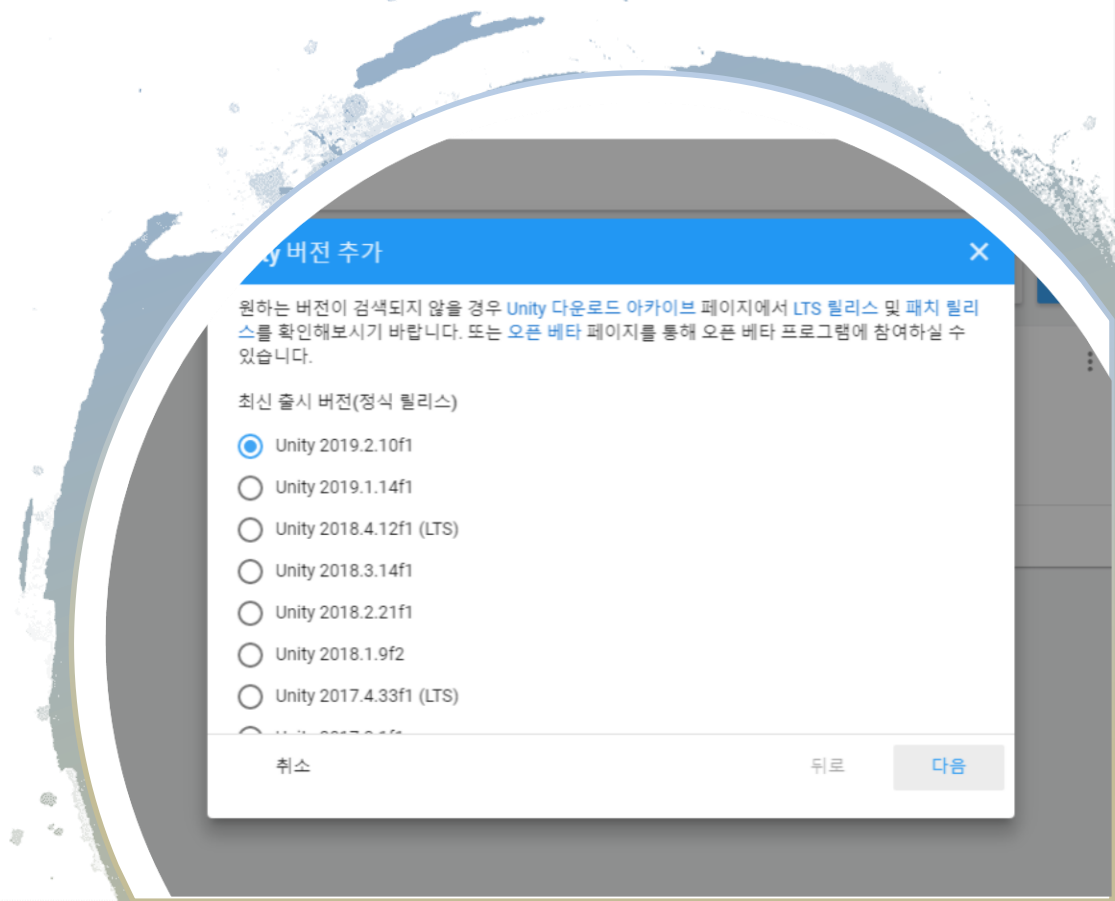


유니티 설치

유니티 사용하기 1

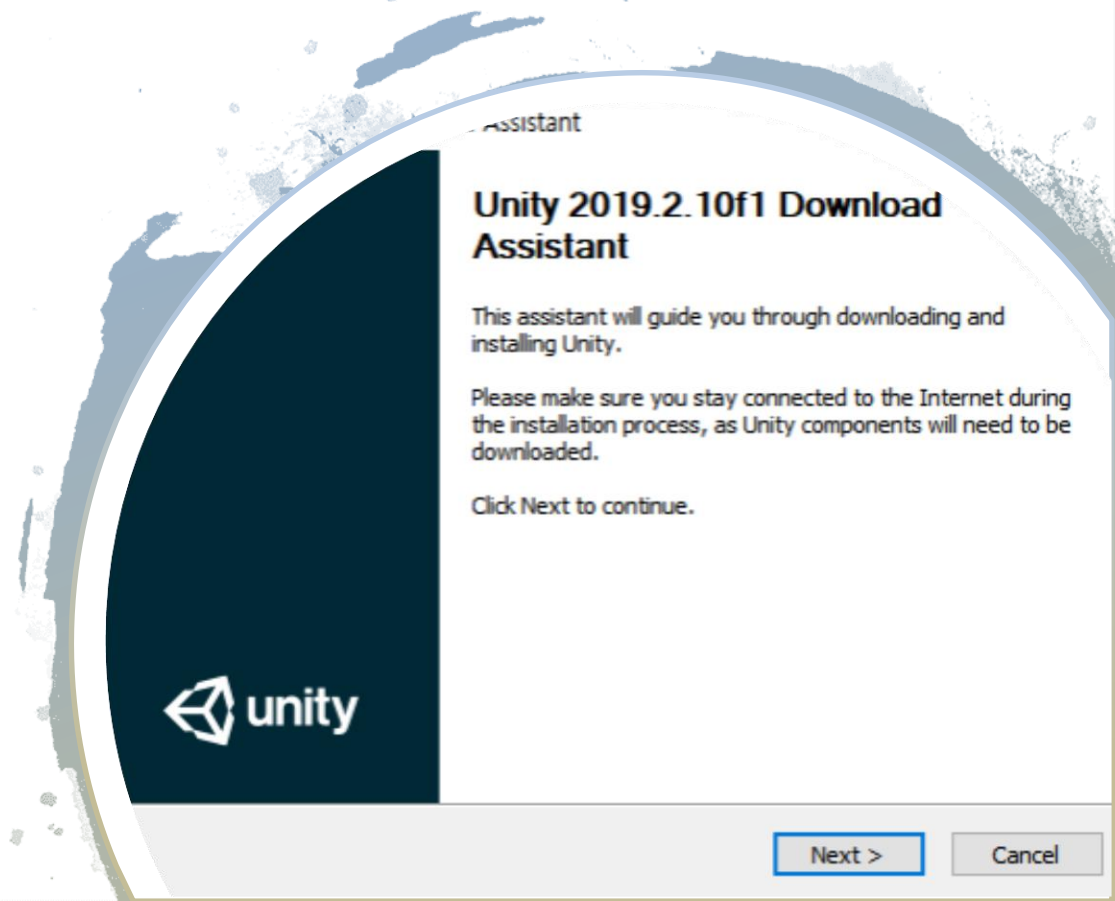
유니티 허브

- 유니티 버전 여러 개 포함



유니티 사용하기 2

유니티 한 버전만 설치



유니티 이전 버전 다운로드

Unity download archive

From this page you can download the previous versions of Unity for both Unity Personal and Pro (if you have a Pro license, enter in your key when prompted after installation). Please note that there is no backwards compatibility from Unity 5; projects made in 5.x will not open in 4.x. However, Unity 5.x will import and convert 4.x projects. We advise you to back up your project before converting and check the console log for any errors or warnings after importing.

Long Term Support releases

The LTS stream is for users who wish to continue to develop and ship their games/content and stay on a stable version for an extended period.

[Download LTS releases](#)

Patch Releases

We are listening to our users who are demanding that we fix more bugs, and faster, with our ongoing patch build releases. Each patch build is a full release of the editor with all runtimes and contains a number of bug fixes.

[Download patch releases](#)

[Unity 2019.x](#) [Unity 2018.x](#) [Unity 2017.x](#) [Unity 5.x](#) [Unity 4.x](#) [Unity 3.x](#)

Unity 2019.2.10
24 Oct, 2019

[Unity Hub](#) [Downloads \(Win\)](#) [Downloads \(Mac\)](#) [Release notes](#)

Unity 2019.2.9
11 Oct, 2019

[Unity Hub](#) [Downloads \(Win\)](#) [Downloads \(Mac\)](#) [Release notes](#)

Unity 2019.2.8

[Unity Hub](#) [Downloads \(Win\)](#) [Downloads \(Mac\)](#) [Release notes](#)

[유니티 아카이브 링크](#)

Unity 버전 추가

원하는 버전이 검색되지 않을 경우 [Unity 다운로드 아카이브](#) 페이지에서 [스](#)를 확인해보시기 바랍니다. 또는 [오픈 베타](#) 페이지를 통해 오픈 베타 있습니다.

최신 출시 버전(정식 릴리스)

- ☒ Unity 2019.2.10f1
- ☐ Unity 2019.1.14f1
- ☐ Unity 2018.4.12f1 (LTS)
- ☐ Unity 2018.3.14f1
- ☐ Unity 2018.2.21f1
- ☐ Unity 2018.1.9f2
- ☐ Unity 2017.4.33f1 (LTS)

취소

장기 지원

장기지원 릴리스(LTS stream)는 지속적으로 게임/콘텐츠를 개발하고 출시하면서 안정적인 버전을 오랫동안 사용하고자 하는 사용자를 위한 릴리스입니다.

장기지원 릴리스에는 새로운 기능과 API 변경 사항이나 개선 사항이 포함되지 않으며, 충돌 문제와 퇴행 문제를 비롯하여 커뮤니티의 다수에게 영향을 미치는 문제, 콘솔 SDK/XDK 관련 문제, 다수의 사용자가 게임을 출시하는 데 지장을 주는 문제를 해결합니다. 각 LTS 스트림은 2년 동안 지원됩니다.

버전: 일반

▼ LTS 릴리스 2018.4.12f1

배포: 28 10월 2019

▼ LTS 릴리스 2018.4.11f1

배포: 11 10월 2019

▼ LTS 릴리스 2017.4.33f1

배포: 7 10월 2019

▼ LTS 릴리스 2018.4.10f1

유니티 종류

일반적인 버전

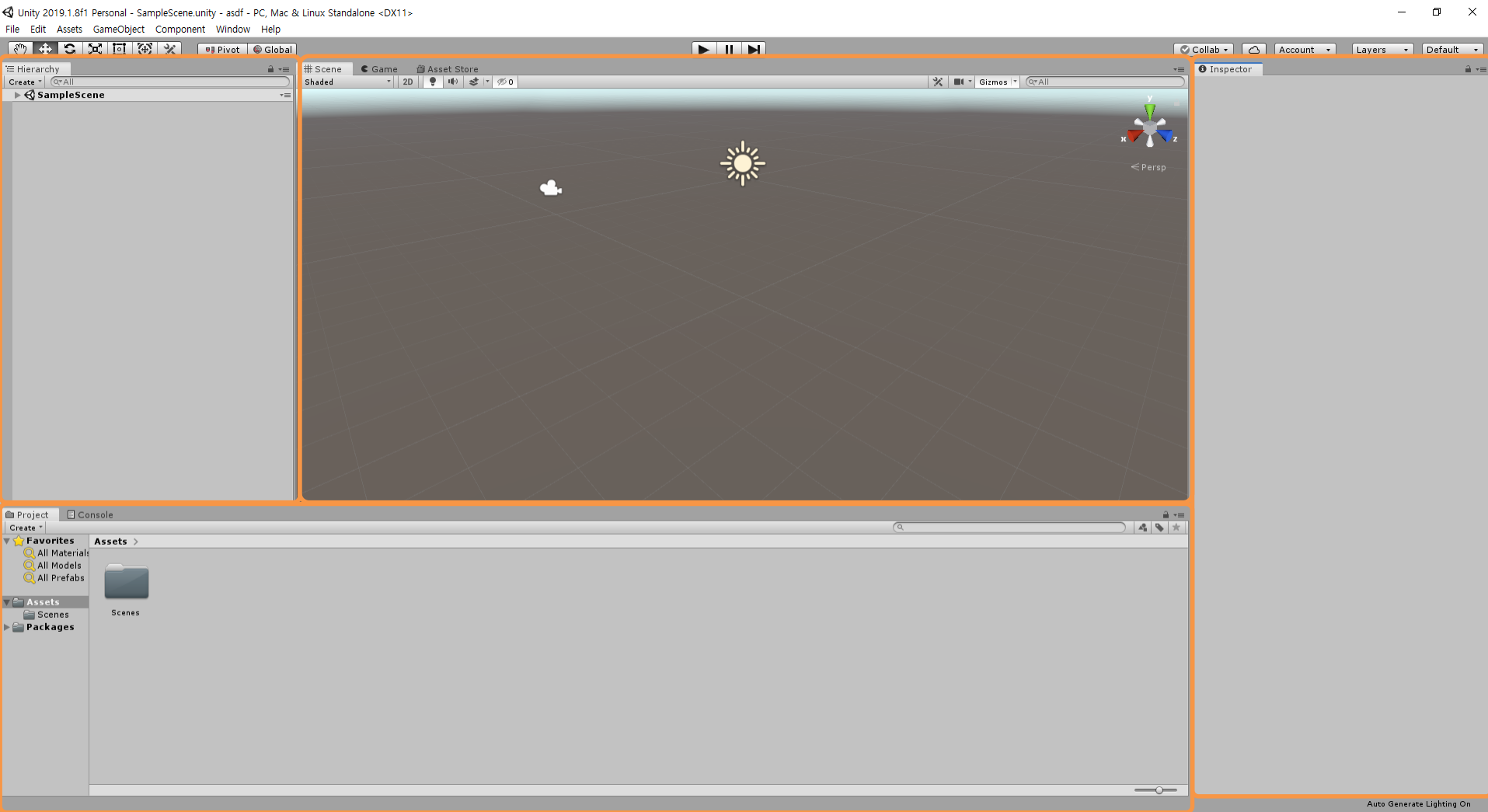
VS

LTS (“Long Term Support”)

#여러 가지의 창들
#프로젝트 #인스펙터
#게임 #씬 #콘솔
#하이라키

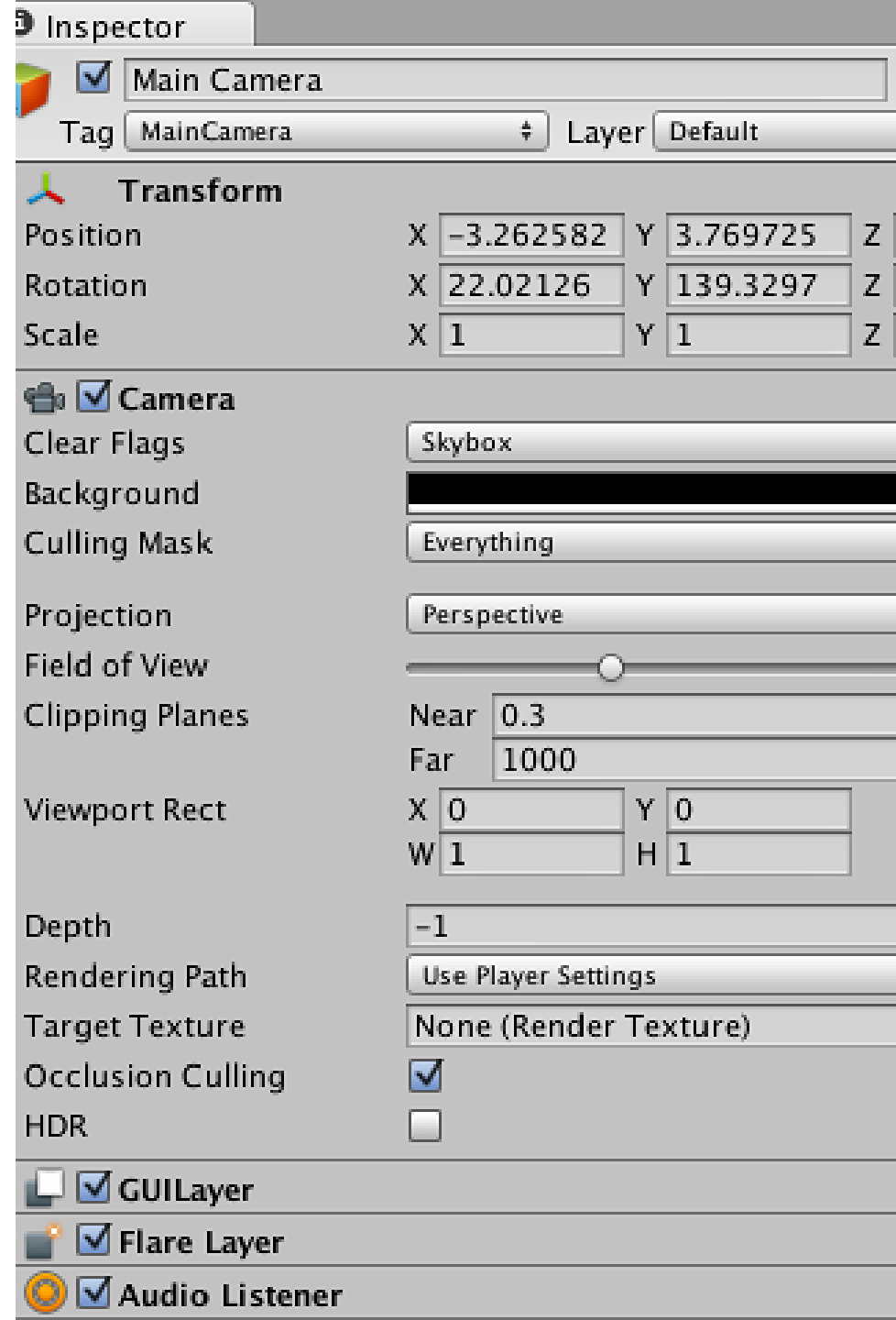
유니티의 화면 구성

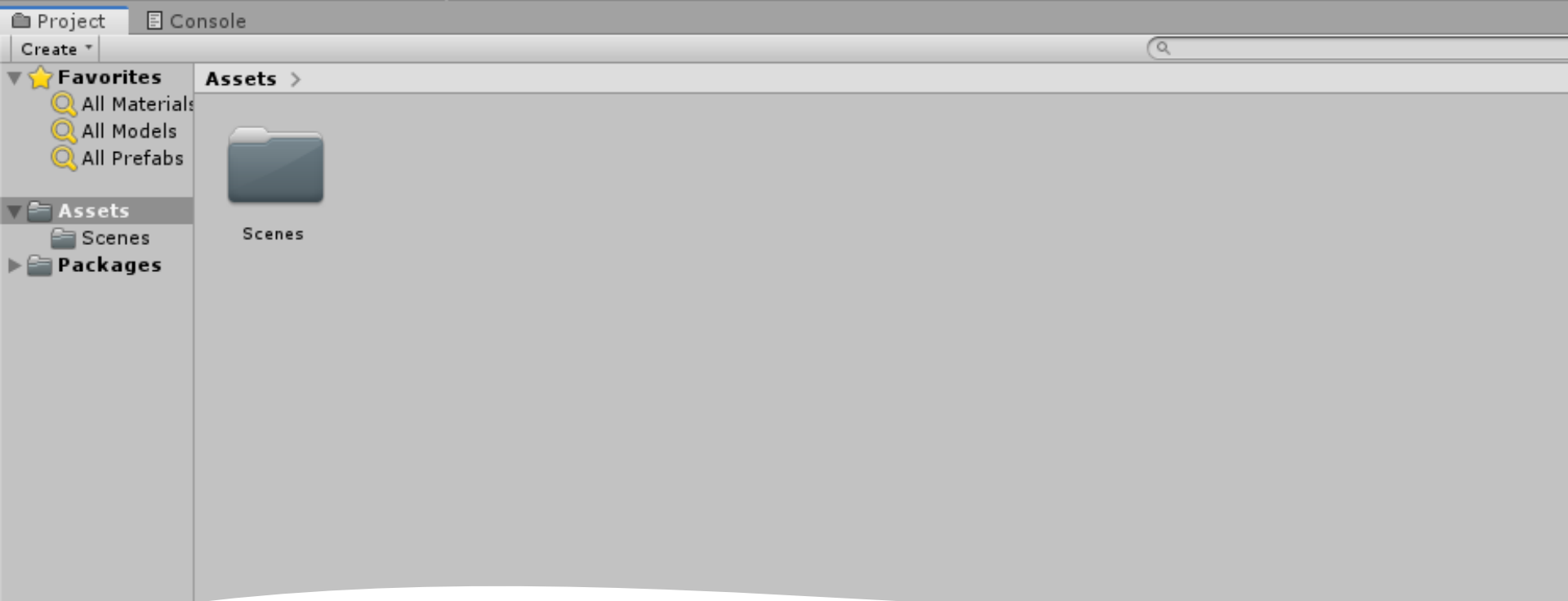
유니티는 여러가지 창으로 구성되어 있습니다



Inspector 창

- Inspecting GameObjects
- 게임 오브젝트들의 컴포넌트들을 볼 수 있다





Project 창

- File 탐색기
- Asset들을 볼 수 있다.

Console 창

- 디버그용
- 메시지를 남길 수 있다
- `Debug.Log(“메시지”);`

Hierarchy 창

- 계층 - 부모자식 관계
- 씬 안의 게임 오브젝트들을 볼 수 있다

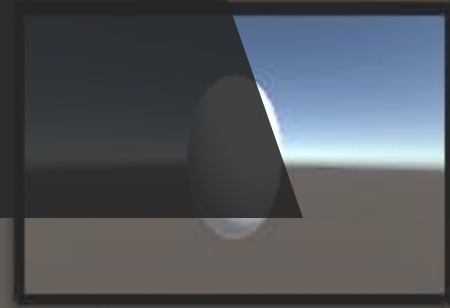




Scene 창

- 내가 구성하고 있는 Scene
- 이 창에서 게임을 만듭니다
=> 게임 오브젝트 이동 및 배치

Camera Preview



Game 창

- Camera가 비추고 있는 장면이 보인다
- 실제 게임을 실행했을 때 보일 창

카메라를 조작해보자

내가 만든 큐브를 비추도록 카메라를
조정해보자

(Tip)
단축키: q, w, e, r, t, y
Focusing: f

조작 방법

게임 오브젝트들을 조작해보자



- QWERTY로 모드 전환
- Scroll
- 우측 마우스 누르고 후 드래그
- 좌측 마우스 누르고 후 드래그
- 게임 오브젝트 누르고 F
- Shift + Arrow
- Ctrl + Arrow

간단한 실습

책상 만들기

-

-

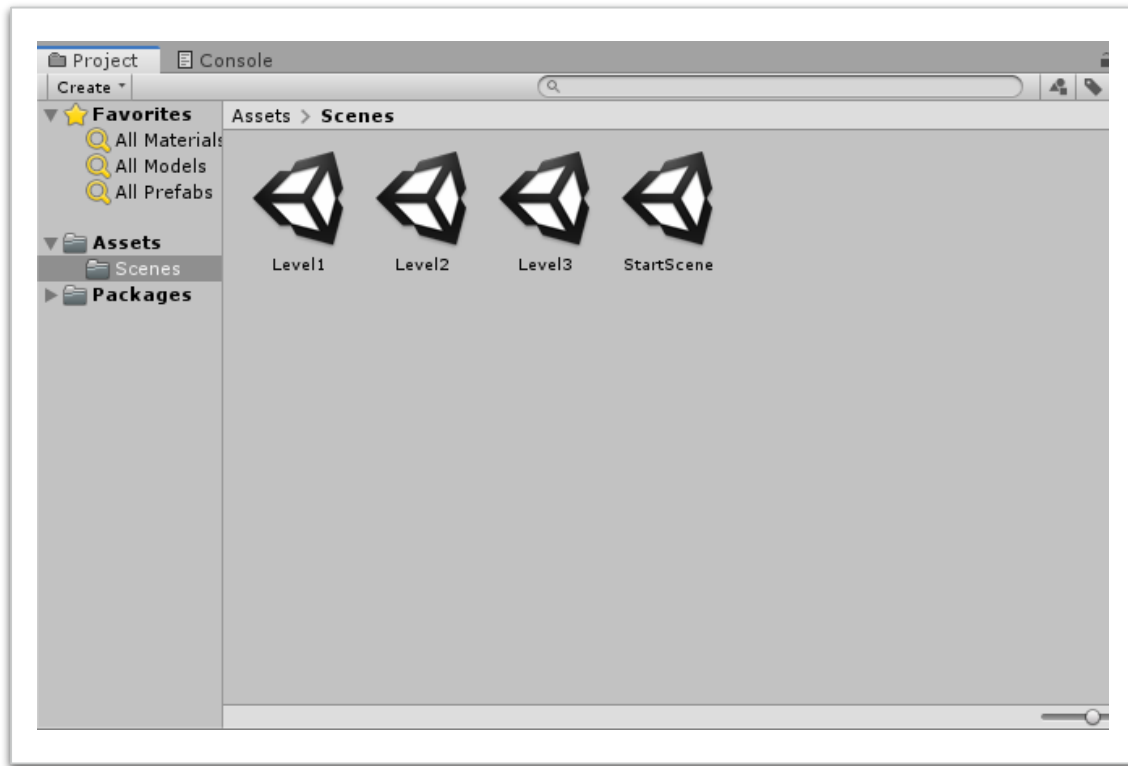
-

-

-

-

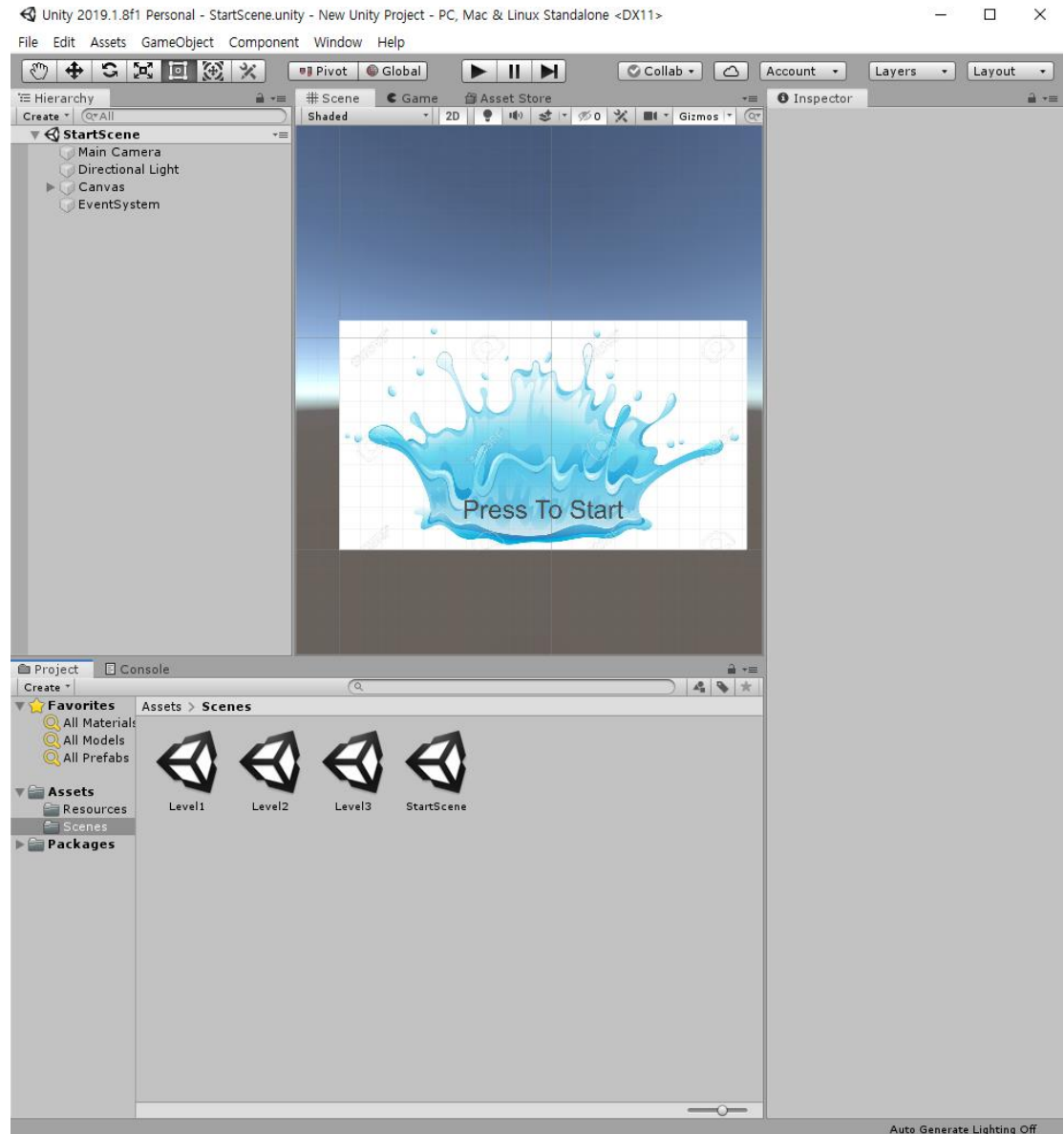
씬 (Scene)



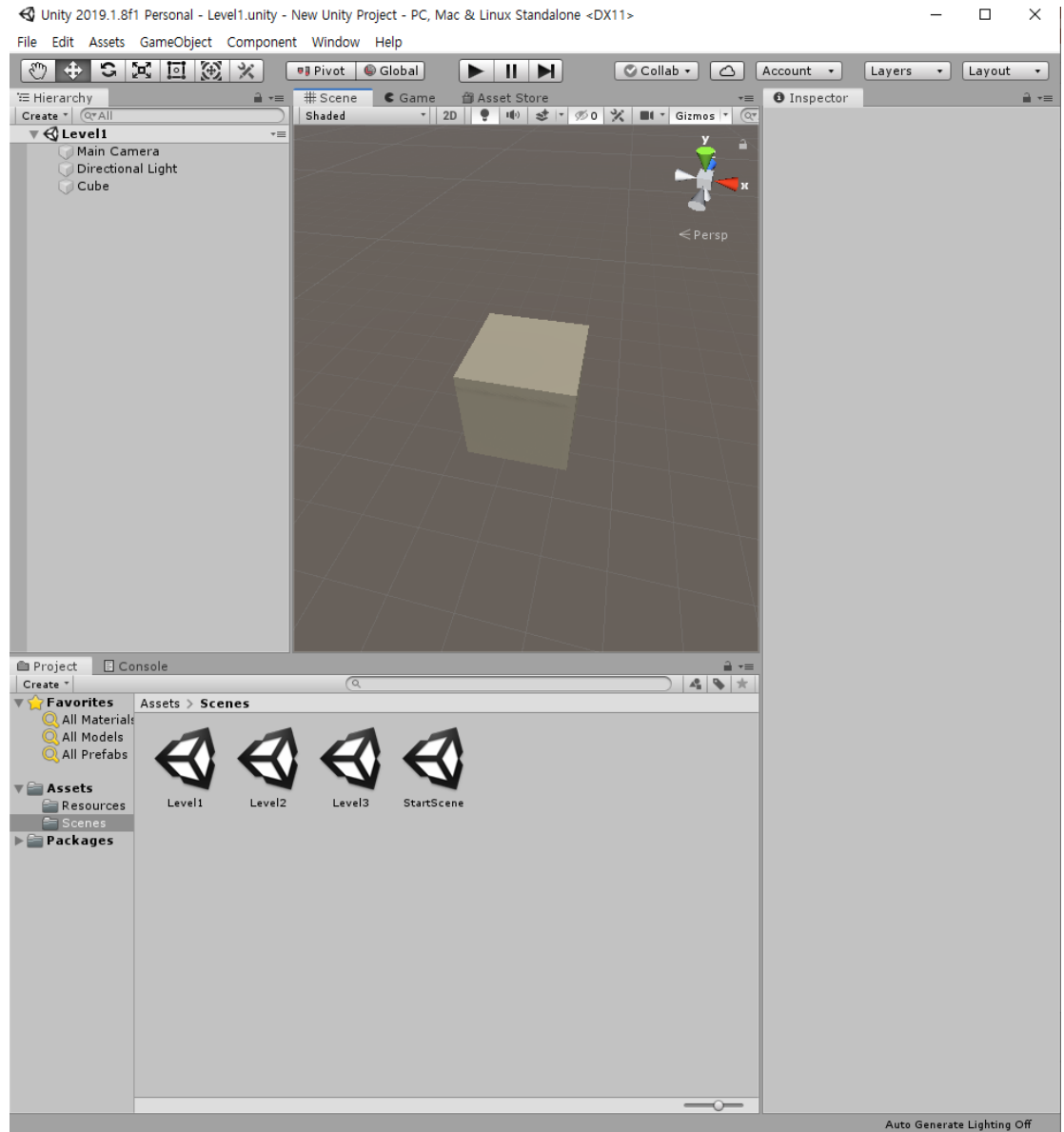
Scene? 장면

= 하나의 레벨/스테이지

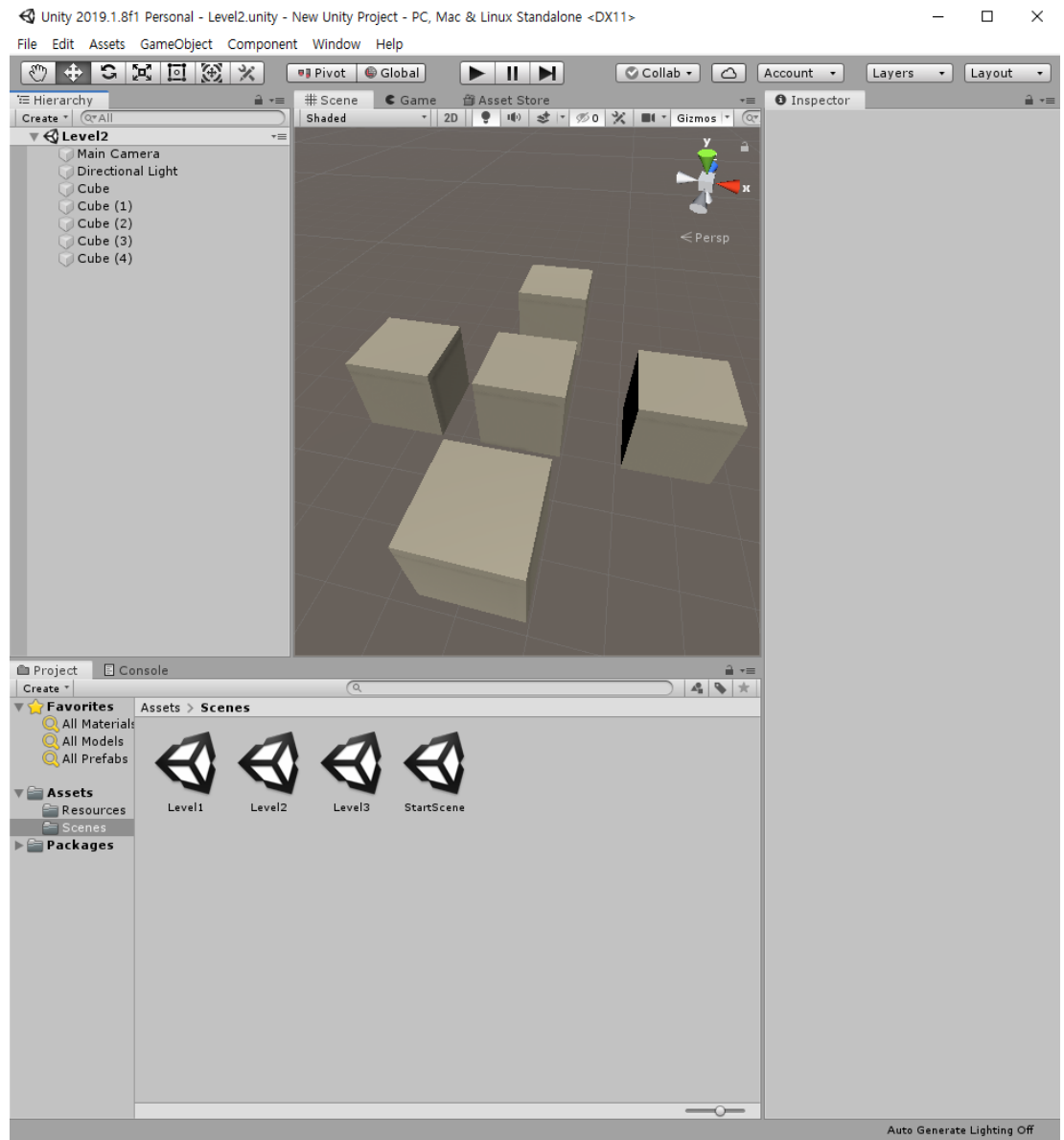
StartScene



Level1



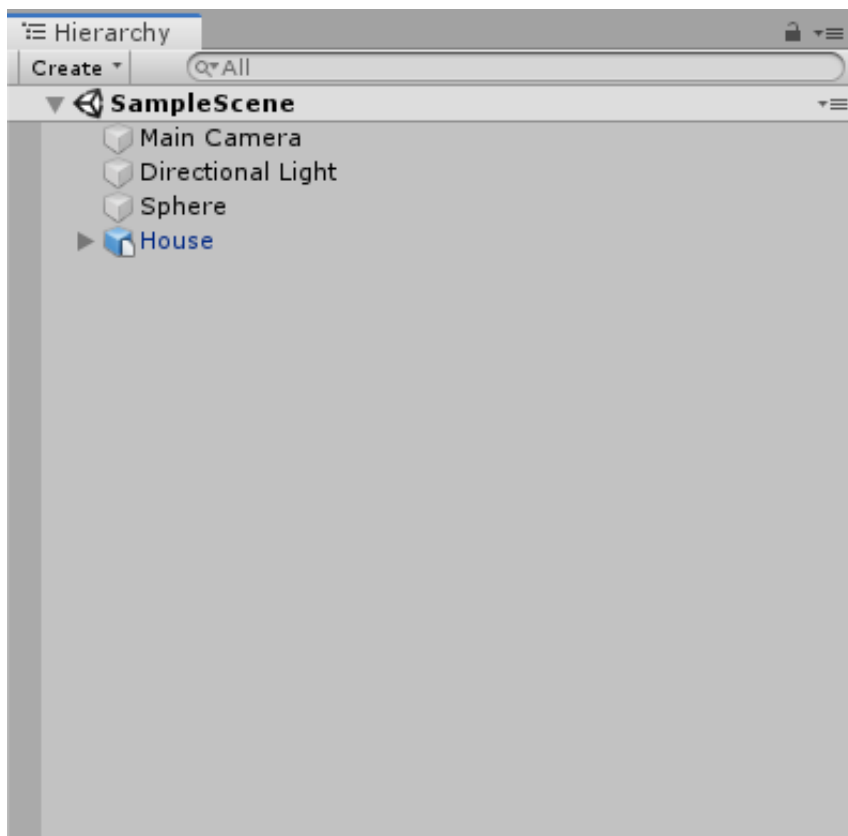
Level2



게임 오브젝트 (GameObject)

GameObject

Hierarchy 창



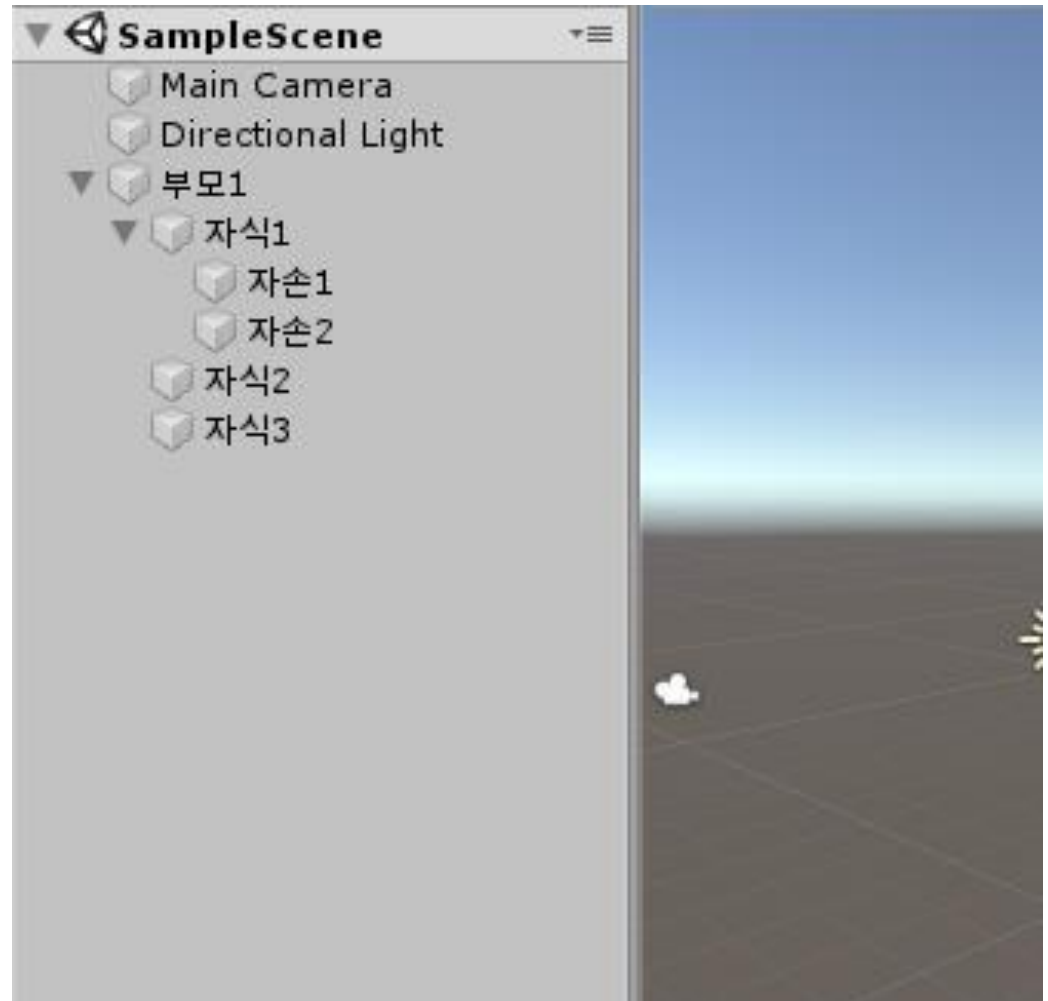
게임 오브젝트란?

- 컴포넌트들의 집합체
 - 컴포넌트들의 컨테이너(상자)
 - Hierarchy 창에서 확인 가능
 - 여러 개의 컴포넌트를 가짐
 - Transform은 항상 가짐
- 왜? World에 존재해야 되기 때문

GameObject

부모-자식 관계

- 게임 오브젝트는 자식을 가질 수 있다
- 상대적:
자손1의 부모는 자식1
자식1,2,3의 부모는 부모1
- 부모에 의존적이다:
부모를 움직이면
자식도 따라 움직인다





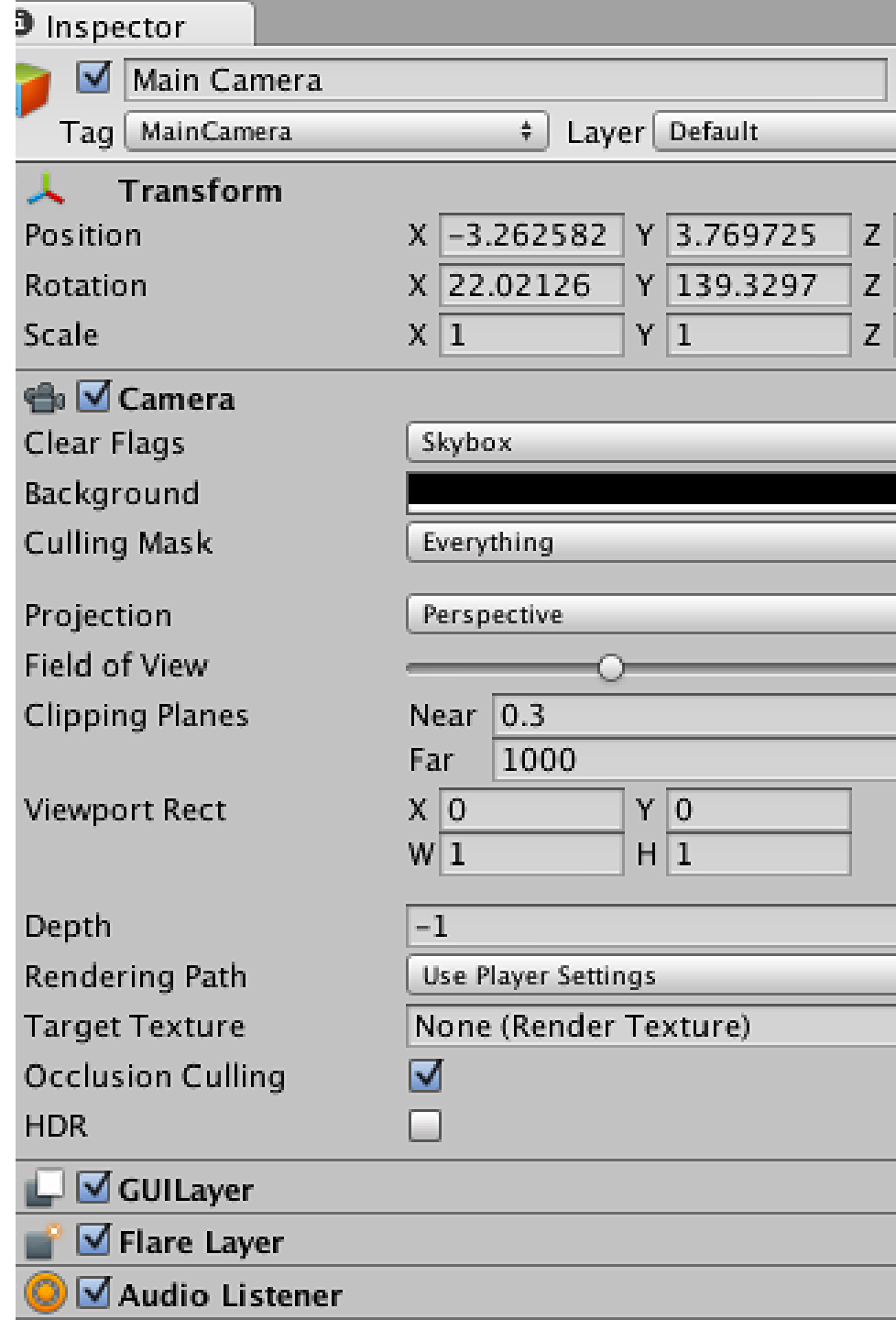
컴포넌트 (Component)

Component

Component란?

게임 오브젝트에 조립할 기능

- 여러가지의 값(변수)으로 구성 됨
- 서로 무관심하다(=서로의 존재를 모른다)
- 게임 오브젝트를 클릭 후 -> Inspector 에서 확인 가능하다



Component 종류

- Collider(콜라이더)
- Camera(카메라)
- Sound Listener(사운드 리스너)
- Transform(트랜스폼)
- Rigidbody(리지드바디)
- Script(스크립트)

공 굴리기

필요한 것?

GameObject => 땅(Plane), 공(Sphere)

Component => 충돌(Collider), 중력(Rigidbody)



스크립트
(Script)

Script

컴포넌트의 청사진이다



```
1 using UnityEngine;
2 using System.Collections;
3 public class NewScript : MonoBehaviour {
4     // Use this for initialization
5     void Start () {
6
7     }
8     // Update is called once per frame
9     void Update () {
10
11     }
12 }
13
```

Script

GameObject의 동작은 연결된 Components에 의해 제어됩니다.

Unity 기본 컴포넌트는 다양한 목적으로 사용할 수 있지만 우리가 게임 기능을 구현하려면 충분하지 않을 때가 많습니다.

Script를 사용하여 사용자 정의 컴포넌트를 생성하면 게임의 이벤트 시작, 시간이 지남에 따라 컴포넌트의 속성을 수정, 사용자 입력 작업에 대한 반응, ...이 가능합니다.

Start 메소드

Update 호출 바로
전에 호출한다

게임 시작할 때 한번만
호출한다

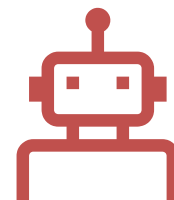
주로 초기화 작업을
해줍니다.

왜 생성자를 안 쓰나요?

오브젝트의 초기화를 할 때 생성자를 안 쓰는 것이 의외일지도 모릅니다.

그 이유는 오브젝트 생성은 에디터에 의해 처리되기 때문입니다. 이 것은 게임 시작 직후에 수행되지 않기 때문에 생성자를 쓰지 않습니다.

스크립트 컴포넌트에서 생성자를 정의하려고 하면 Unity의 일반적 처리에 간섭하기 때문에 프로젝트에서 문제를 발생시킵니다.



Update 메소드

매 프레임마다
호출한다

FPS(Frame
Per Second)

Debug 찍기

- 콘솔창에 “메시지” 출력한다
- 왜? 디버그 용 / 확인 용

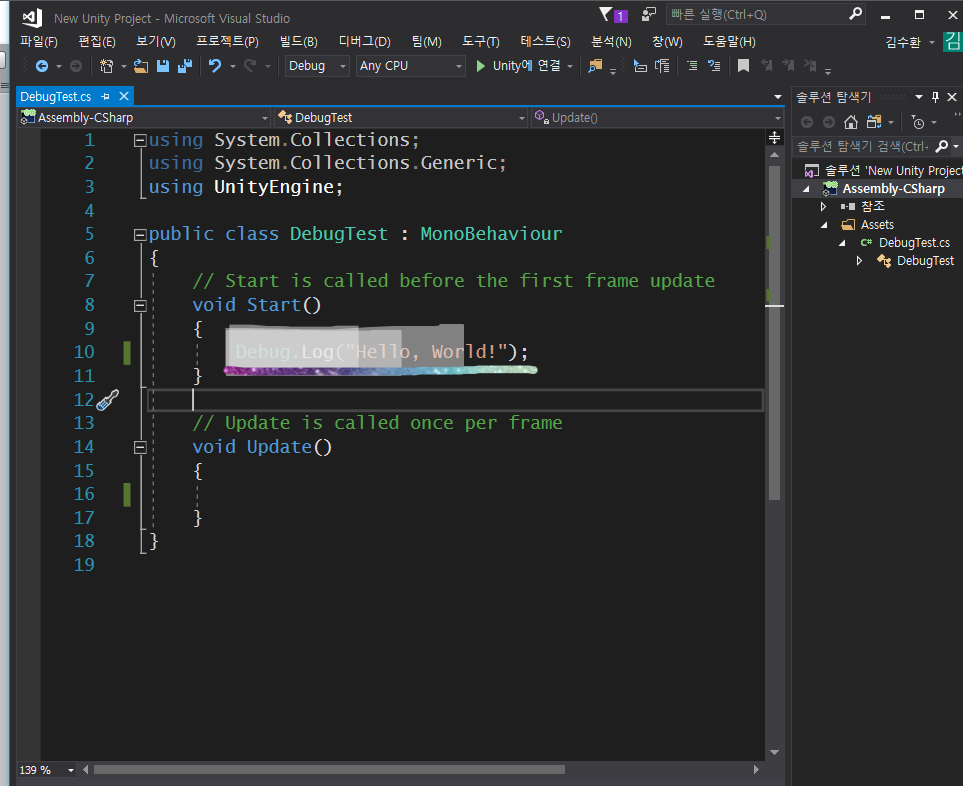
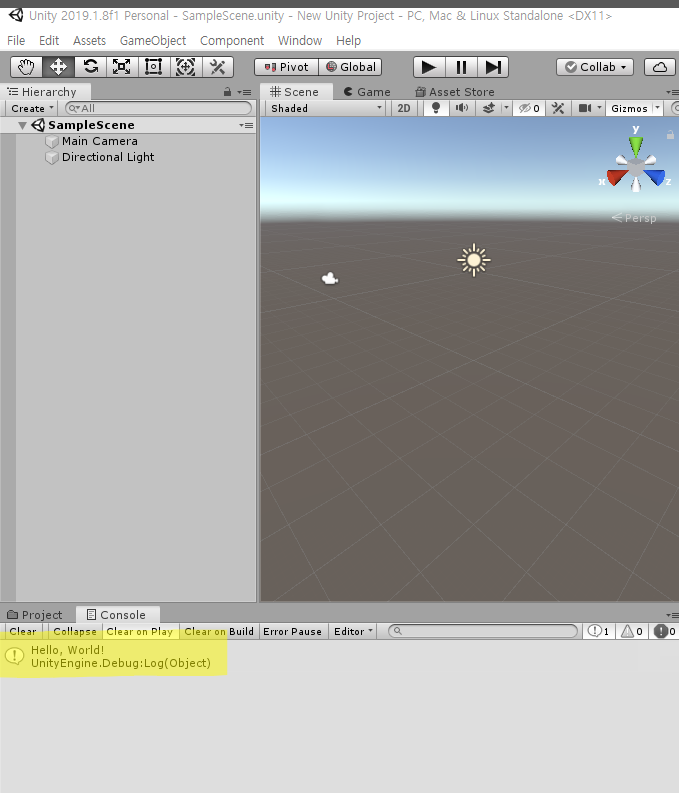
```
Debug.Log(“메시지”);
```

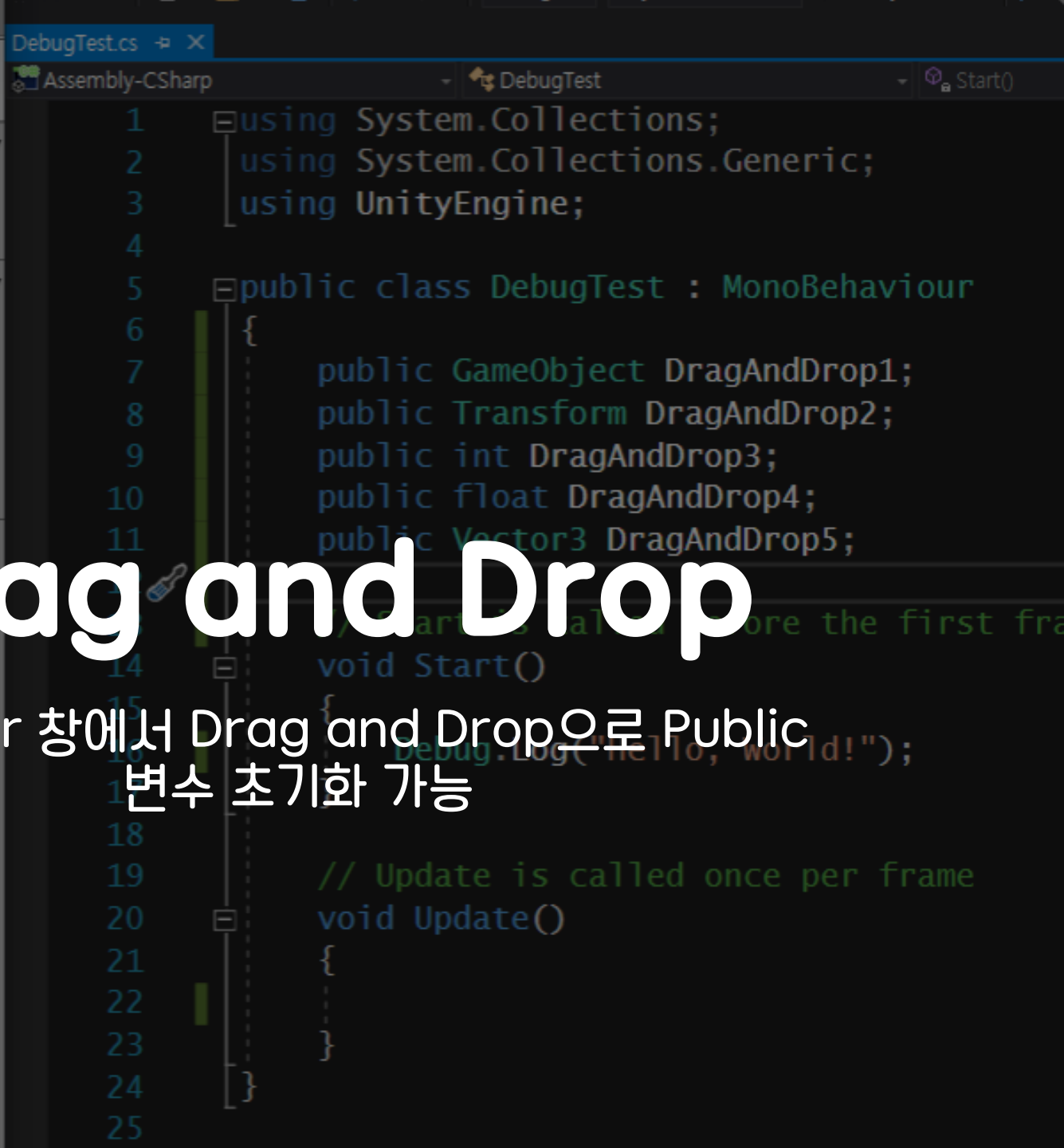
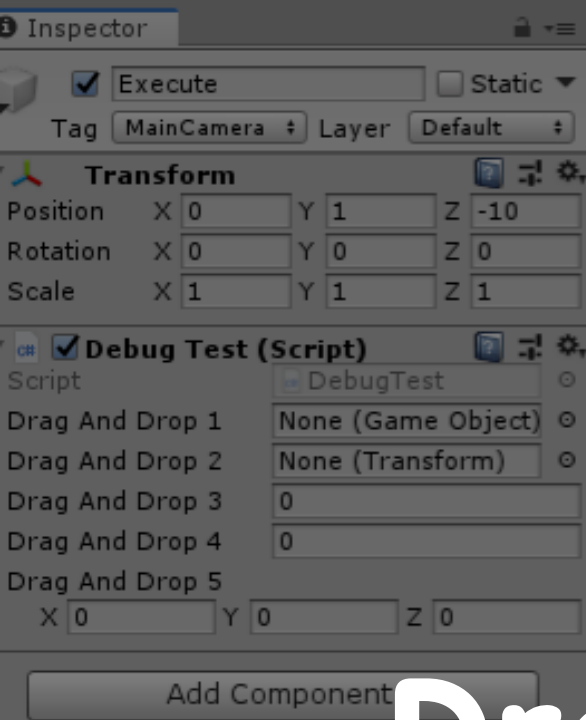
\$ 메시지

```
Debug.Log(변수);
```

\$ 변수값 출력

스크립트





Drag and Drop

Inspector 창에서 Drag and Drop으로 Public 변수 초기화 가능

Inspector

☒ Execute ☐ Static

Tag MainCamera Layer Default

Transform

Position X 0 Y 1 Z -10

Rotation X 0 Y 0 Z 0

Scale X 1 Y 1 Z 1

Debug Test (Script)

Script DebugTest

Drag And Drop 1 None (Game Object)

Drag And Drop 2 None (Transform)

Drag And Drop 3 0

Drag And Drop 4 0

Drag And Drop 5 X 0 Y 0 Z 0

Add Component

DebugTest.cs

```
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4
5 public class DebugTest : MonoBehaviour
6 {
7     public GameObject DragAndDrop1;
8     public Transform DragAndDrop2;
9     public int DragAndDrop3;
10    public float DragAndDrop4;
11    public Vector3 DragAndDrop5;
12
13    // Start is called before the first frame update
14    void Start()
15    {
16        Debug.Log("Hello, World!");
17    }
18
19    // Update is called once per frame
20    void Update()
21    {
22    }
23
24 }
25
```

Inspector

☒ Execute ☐ Static

Tag MainCamera Layer Default

Transform

Position	X	0	Y	1	Z	-10
Rotation	X	0	Y	0	Z	0
Scale	X	1	Y	1	Z	1

☒ **Debug Test (Script)**

Script DebugTest

Drag And Drop 1	Execute
Drag And Drop 2	Execute (Transform)
Drag And Drop 3	111
Drag And Drop 4	222
Drag And Drop 5	

X	1	Y	1	Z	2
---	---	---	---	---	---

Add Component

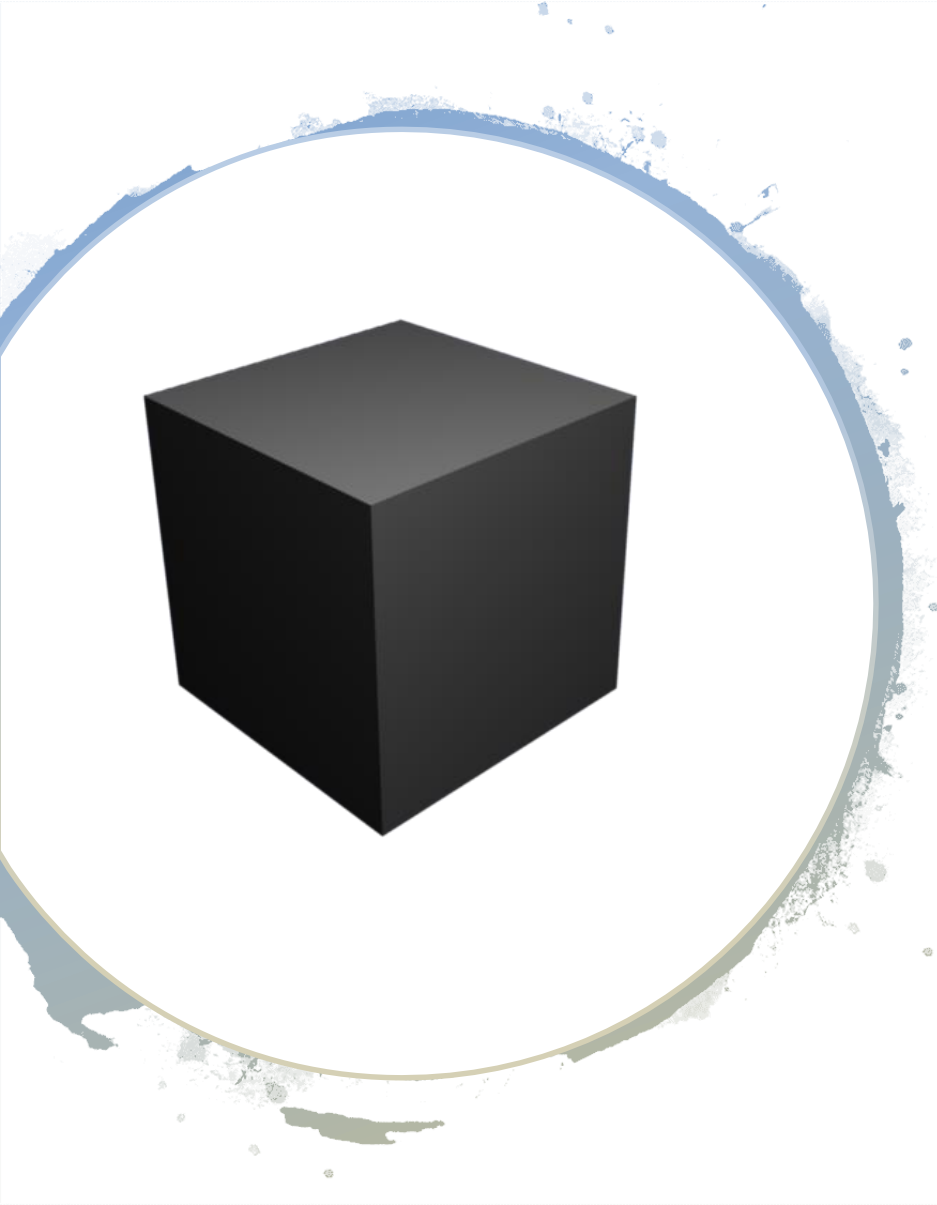
DebugTest.cs

Assembly-CSharp

DebugTest

Update

```
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4
5 public class DebugTest : MonoBehaviour
6 {
7     public GameObject DragAndDrop1;
8     public Transform DragAndDrop2;
9     public int DragAndDrop3;
10    public float DragAndDrop4;
11    public Vector3 DragAndDrop5;
12
13    // Start is called before the first
14    void Start()
15    {
16        Debug.Log("Hello, World!");
17    }
18
19    // Update is called once per frame
20    void Update()
21    {
22    }
23 }
24
25
```



Input.GetKey(KeyCode) 함수

**화살표 키로 Cube를 상하좌우로 움직이는
Script를 만들어 보자!**



`Input.GetKey(키값)`

만약에 KeyCode(키값)에 해당하는 Key가 눌리면
“True”

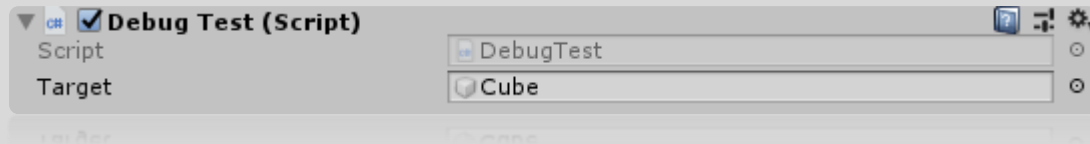
아니면
“False”

1. Target GameObject를 설정하자

스크립트 안에서 변수 선언

```
public GameObject target;
```

인스펙터 창으로 돌아가서 Drag and Drop으로 target설정



2. 키보드 입력을 받아서 움직이자

스크립트 안에서

```
// Update is called once per frame
void Update()
{
    // 상
    if (Input.GetKey(KeyCode.W))
    {
        target.transform.Translate(0, 0, 0.1f);
    }
    // 하
    if (Input.GetKey(KeyCode.S))
    {
        target.transform.Translate(0, 0, -0.1f);
    }
    // 좌
    if (Input.GetKey(KeyCode.A))
    {
        target.transform.Translate(-0.1f, 0, 0);
    }
    // 우
    if (Input.GetKey(KeyCode.D))
    {
        target.transform.Translate(0.1f, 0, 0);
    }
}
```

Full Code

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

public class DebugTest : MonoBehaviour
{
    public GameObject target;

    // Start is called before the first frame update
    void Start()
    {
    }

    // Update is called once per frame
    void Update()
    {
        Debug.Log(Input.GetKey(KeyCode.A));
        // 상
        if (Input.GetKey(KeyCode.W))
        {
            target.transform.Translate(0, 0, 0.1f);
        }
        // 하
        if (Input.GetKey(KeyCode.S))
        {
            target.transform.Translate(0, 0, -0.1f);
        }
        // 좌
        if (Input.GetKey(KeyCode.A))
        {
            target.transform.Translate(-0.1f, 0, 0);
        }
        // 우
        if (Input.GetKey(KeyCode.D))
        {
            target.transform.Translate(0.1f, 0, 0);
        }
    }
}
```

Frame?

```
Update(){  
    // 대충 1미터 움직이라는 코드  
}
```

으악 프레임 드랍!

11	frame/sec => 1초에 11미터 이동
60	frame/sec => 1초에 60미터 이동

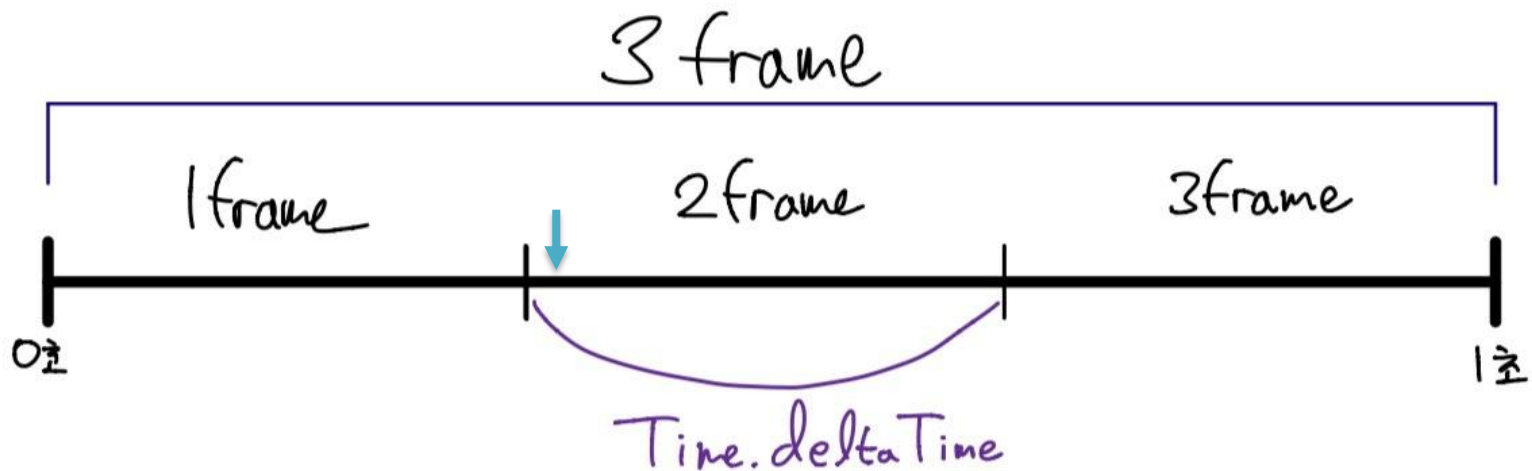
해결법

“Time.deltaTime”을 곱해준다

- “Time.deltaTime”이란?
 - 지난 프레임이 완료되는 데 까지 걸린 시간
 - 단위: 초
- “Speed * Time.deltaTime”
 - 1초마다 speed만큼 이동한다
 - 프레임 별로 속도 차이가 없어진다

Time.deltaTime

3 fps = 1초에 3 frame



출처



유니티 매뉴얼

<https://docs.unity3d.com/Manual/index.html>