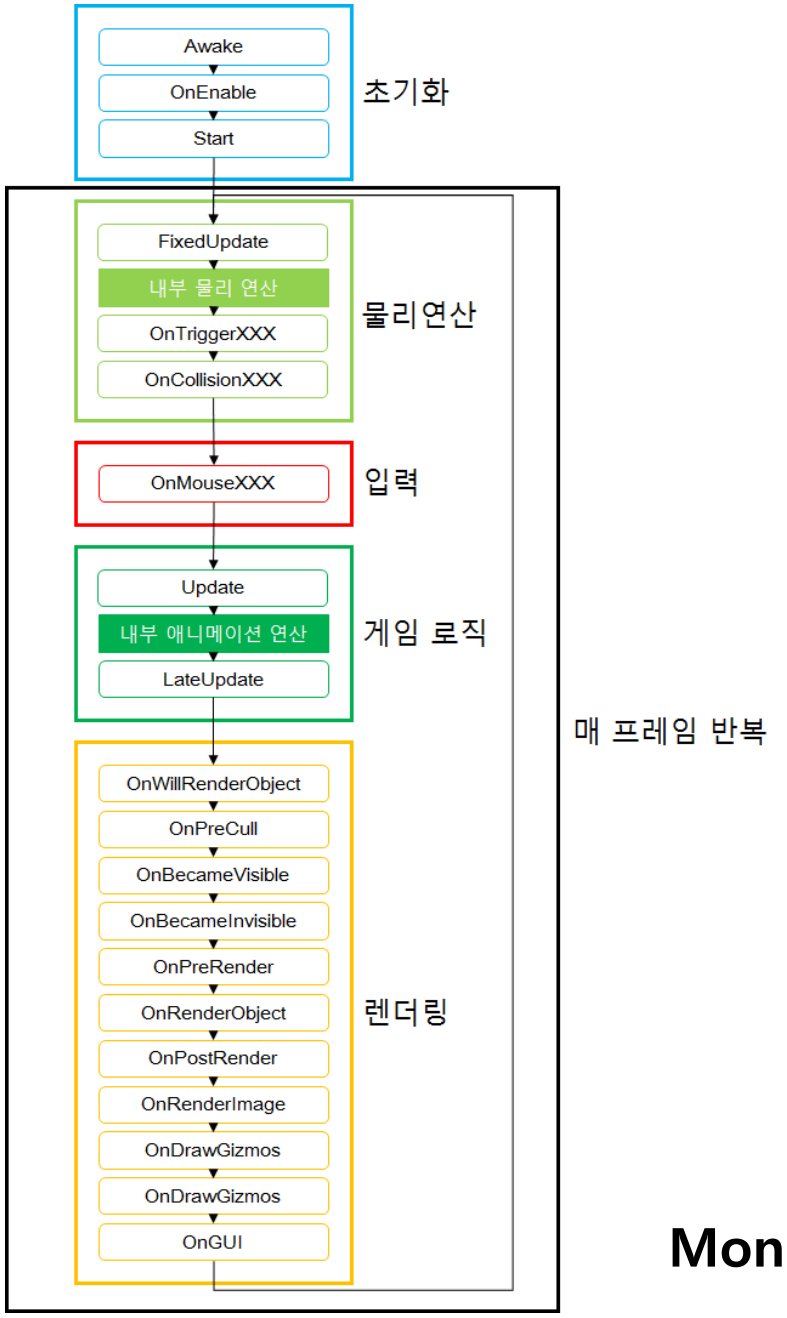
유니티 엔진 클래스

MONOBEHAVIOR 클래스

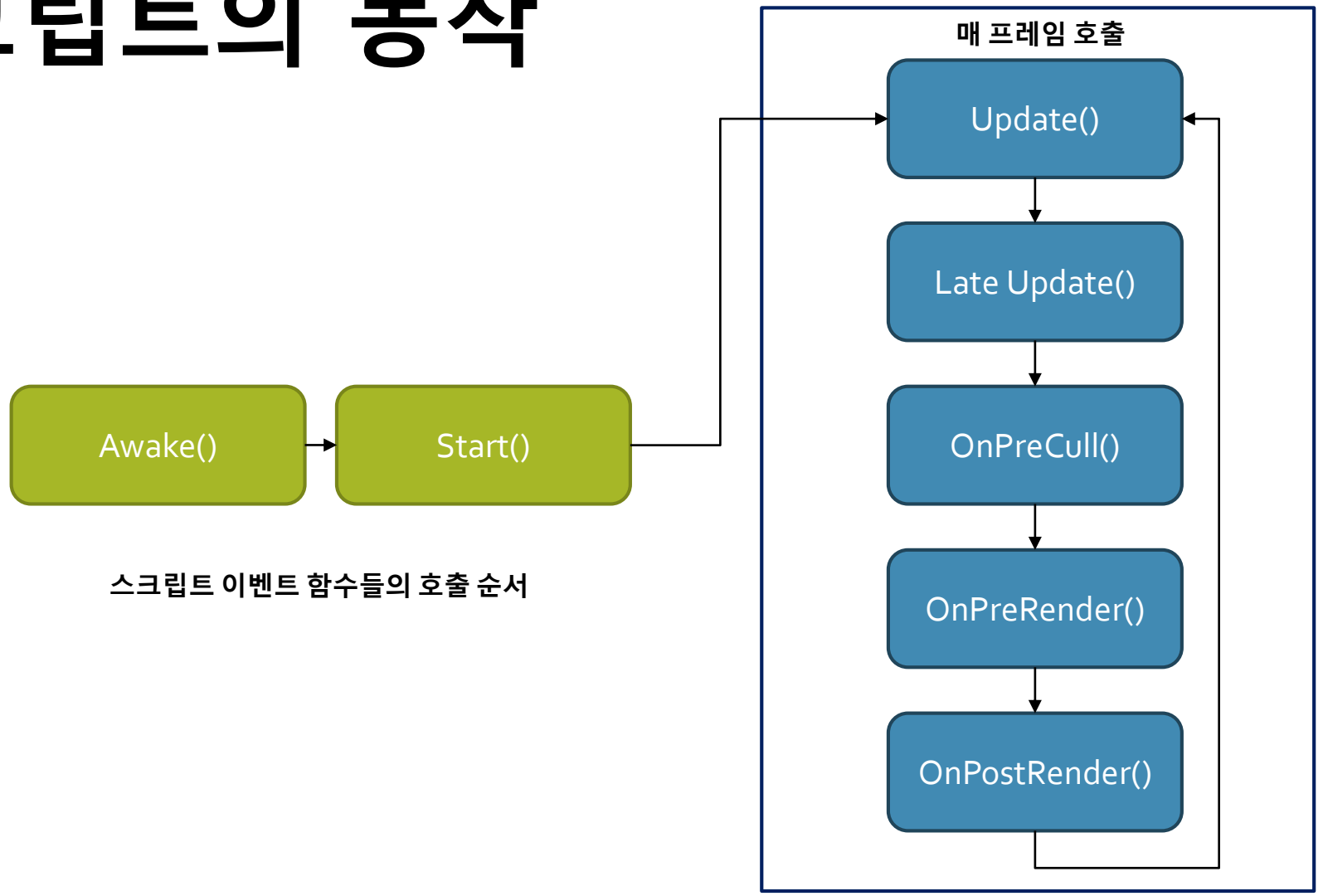
MonoBehavior 클래스

- 유니티 스크립트는 MonoBehavior 클래스를 상속받아 시작한다.
- 구현 내용
  - 스크립트가 실행되면서 발생하는 이벤트를 처리하기 위한 메소드 구현
  - 게임 오브젝트의 생성과 소멸 검색등을 처리하는 메소드 구현
  - 스크립트를 사용하는 게임 오브젝트, 게임 오브젝트의 컴포넌트에 접근하기 위한 필드를 제공



MonoBehavior 이벤트 처리 과정

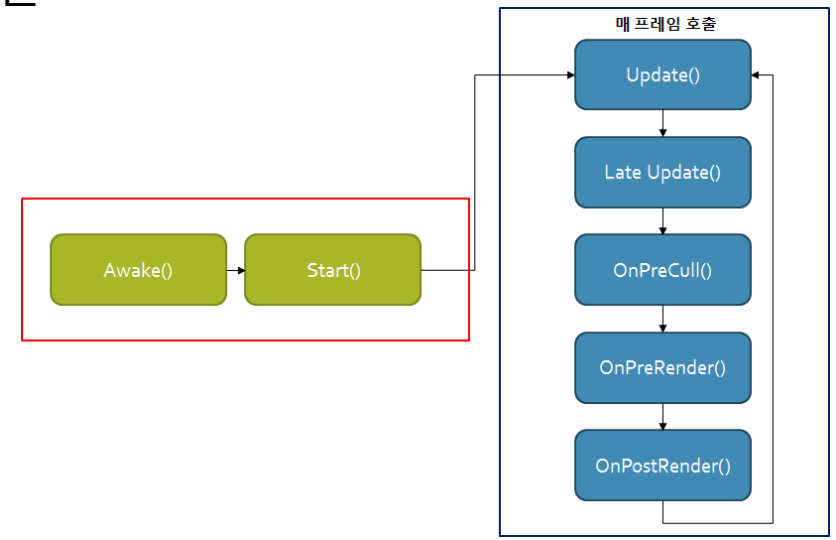
스크립트의 동작



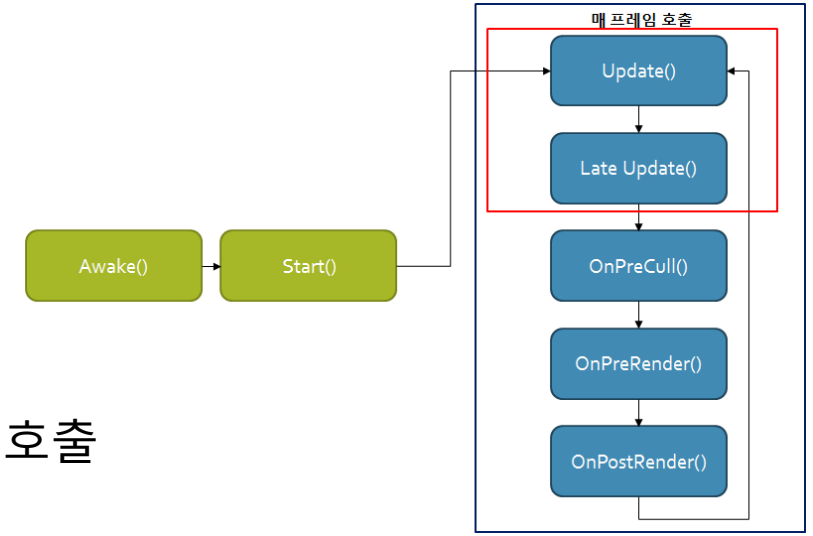
스크립트 이벤트 함수들의 호출 순서

스크립트의 동작

- **Awake()**
  - 스크립트가 붙은 게임 오브젝트가 생성되면 단 한번 호출
  - Start() 함수와 같이 초기화 함수로 사용
  - Awake() 함수 실행 후 Start() 함수 실행
- **Start()**
  - 프레임 렌더링 루프에 들어가기 전 한번 실행
  - 초기화 함수로 사용



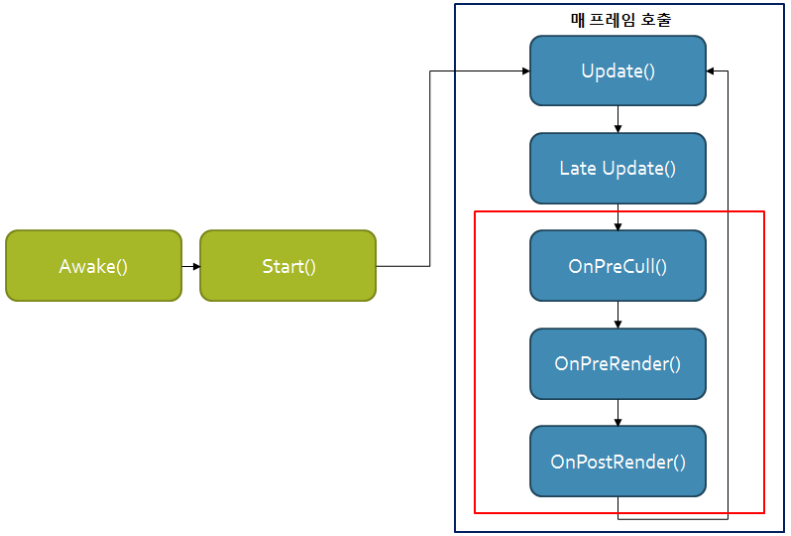
스크립트의 동작



- **Update()**
  - 매 프레임 카메라가 게임 오브젝트들을 렌더링 하기 전 호출
  - 주로 게임의 로직과 상태 변화를 처리
  - 물리 시뮬레이션 관련 코드는 처리하지 않도록 한다.**(FixedUpdate() 에서 처리)**
- **LateUpdate()**
  - Update() 수행 후 호출

스크립트의 동작

- **OnPreCull()**
  - 카메라가 씬을 컬링하기 전 호출
- **OnPreRender()**
  - 카메라가 화면을 그리기 전에 호출
- **OnPostRender()**
  - 카메라가 화면을 그린 후 호출



컴포넌트 접근

- 컴포넌트에 접근하는 방법
  - **GetComponent<컴포넌트 명> ()**

    Ex) GetComponent<Transform> ().Rotate (0.0f, speed * Time.deltaTime, 0.0f);
  - 컴포넌트 필드 이용

    Ex) transform.Rotate (0.0f, speed * Time.deltaTime, 0.0f);

다른 게임 오브젝트에 접근

- 게임 오브젝트 명으로 검색
- Inspector View를 이용하여 스크립트 연결
- 계층 구조 내 자식 노드 검색
- 이름과 태그를 통해 접근

게임 오브젝트 명으로 검색

- 멤버 변수에 접근
  - GetComponent<게임 오브젝트명>().멤버 변수
- 멤버 함수에 접근
  - GetComponent<게임 오브젝트명>().멤버 함수(...)

```
1  using UnityEngine;
2  using System.Collections;
3
4  public class Undead : MonoBehaviour {
5      void Awake()
6      {
7          transform.Find ("Zombie").GetComponent<Zombie> ().Answer ();
8          transform.Find ("Skeleton").GetComponent<Skeleton> ().Answer ();
9      }
10
11     public void Answer(){
12         Debug.Log ("We are Undead");
13     }
14 }
15
```

멤버 함수에 접근

Inspector View를 이용하여 스크립트 연결

- 연결할 클래스를 **public** 멤버 변수로 선언 후 Inspector View 해당 스크립트를 드래그&드랍
- 선언한 멤버 변수의 타입에 맞추어 자동으로 컴포넌트를 가져올 수 있다.

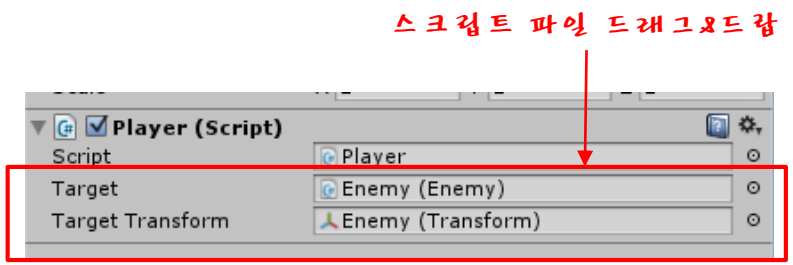
```
1 using UnityEngine;
2 using System.Collections;
3
4 public class Player : MonoBehaviour {
5     public Enemy target;
6     public Transform targetTransform;
7
8     // Use this for initialization
9     void Start () {
10        target.Attack ();
11        Debug.Log (targetTransform.position);
12    }
13
14    void Update() {
15        target.Idle ();
16    }
17 }
```

멤버 변수 선언

Player.cs

```
1 using UnityEngine;
2 using System.Collections;
3
4 public class Enemy : MonoBehaviour {
5
6     public void Attack() {
7         Debug.Log ("Debug:Attack");
8     }
9
10    public void Idle() {
11        Debug.Log ("Debug::Idle");
12    }
13 }
14
```

Enemy.cs



Inspector View 에 스크립트 연결

계층 구조 내 자식 노드 검색

- 트랜스폼(**Transform**) 컴포넌트의 **Find** 함수를 사용하여 자식 노드 검색

```
1  using UnityEngine;
2  using System.Collections;
3
4  public class Enemy : MonoBehaviour {
5      void Awake() {
6          transform.Find ("Undead").GetComponent<Undead> ().Answer ();
7      }
8
9      void Start() {
10         Answer ();
11     }
12
13     public void Answer()
14     {
15         Debug.Log ("We are Enemy");
16     }
17 }
18
```

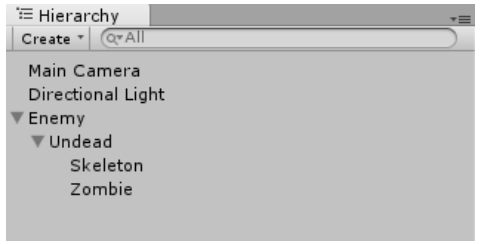
자식 노드 검색 후 접근

Enemy.cs

호출

```
1  using UnityEngine;
2  using System.Collections;
3
4  public class Undead : MonoBehaviour {
5      void Awake()
6      {
7          transform.Find ("Zombie").GetComponent<Zombie> ().Answer ();
8          transform.Find ("Skeleton").GetComponent<Skeleton> ().Answer ();
9      }
10
11     public void Answer(){
12         Debug.Log ("We are Undead");
13     }
14 }
15
```

Undead.cs



게임 오브젝트 계층 구조

이름과 태그를 통해 접근

- 게임 오브젝트의 이름으로 찾고자 할 때
  - **GameObject.Find(“클래스명”)**
- 게임 오브젝트의 태그로 찾고자 할 때
  - **GameObject.FindWithTag(“태그명”)**

```
1  using UnityEngine;
2  using System.Collections;
3
4  public class Player : MonoBehaviour {
5
6      // Use this for initialization
7      void Start () {
8          GameObject skeleton = GameObject.Find ("Skeleton");
9          skeleton.GetComponent<Skeleton> ().Answer ();
10         GameObject terribleZombie = GameObject.FindWithTag ("terrible");
11         terribleZombie.GetComponent<Zombie> ().Answer ();
12     }
13 }
```

이름 검색

태그 검색

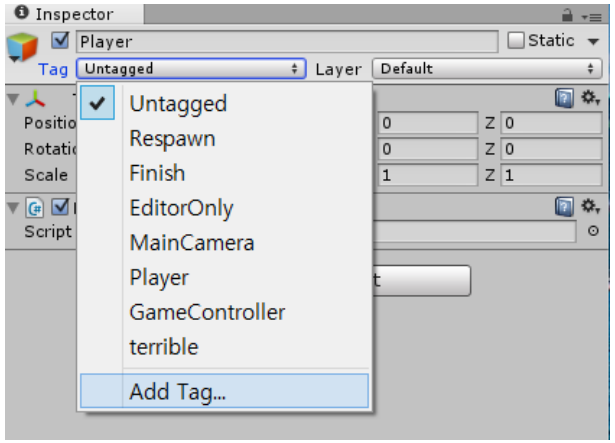
Player.cs

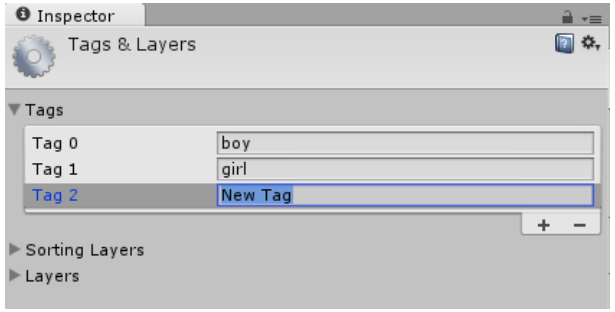
태그의 추가

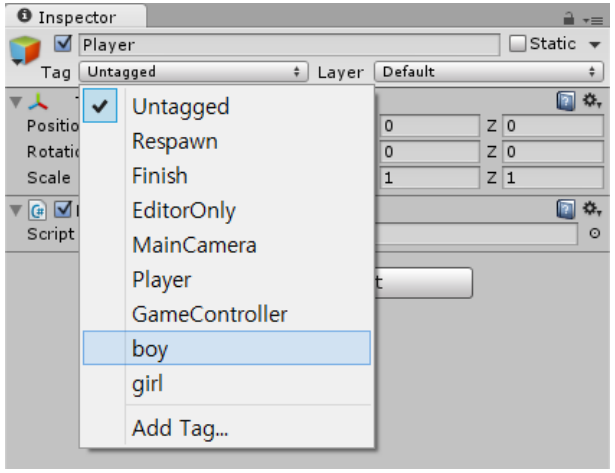
① Inspector View>> Tag 드롭다운박스 >> Add Tag…

② 설정할 Tag 이름들 추가

③ Inspector View>> Tag 드롭다운박스 >> Tag 선택







게임 오브젝트의 생성과 소멸

- **게임 오브젝트의 생성**
  - public static Object **Instantiate**(Object original, Vector3 position, Quaternion rotation);
  - public static Object **Instantiate**(Object original);
- **게임 오브젝트의 소멸**
  - public static void **Destroy**(Object obj, float t = 0.0F);

INPUT 클래스

입력 처리

- Input 클래스
  - 유니티의 어플리케이션의 입력 시스템의 인터페이스
  - 각종 입력에 따른 수치 데이터를 얻을 수 있다

- Input.GetAxis

  Input.GetAxis("Vertical")

  Input.GetAxis("Horizontal")

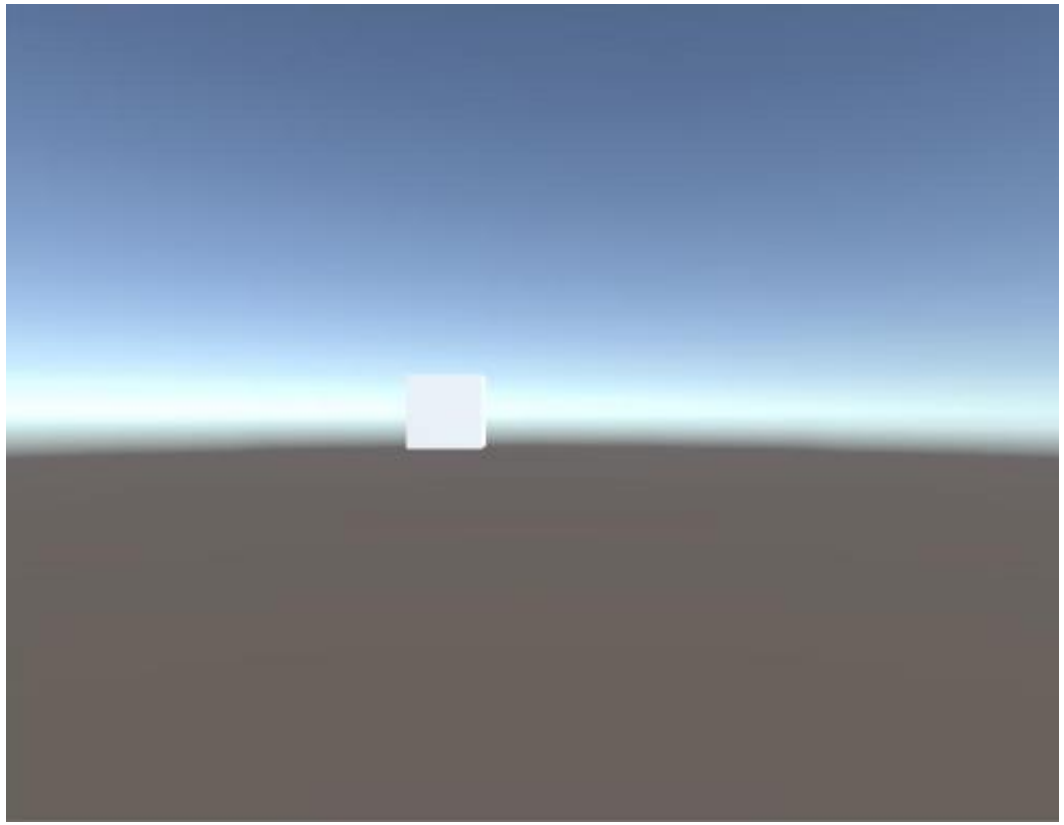
  Input.GetAxis("Mouse X")

  Input.GetAxis("Mouse Y")
- Input.GetKey

  Input.GetKey(KeyCode.Space)
- Input.GetKeyDown
- Input.GetKeyUp

직접 해 봅시다

- 키보드 좌우 키를 누르면 큐브가 좌우로 이동
- 스페이스 키를 누르면 콘솔창에 “Shoot” 출력



TIME 클래스

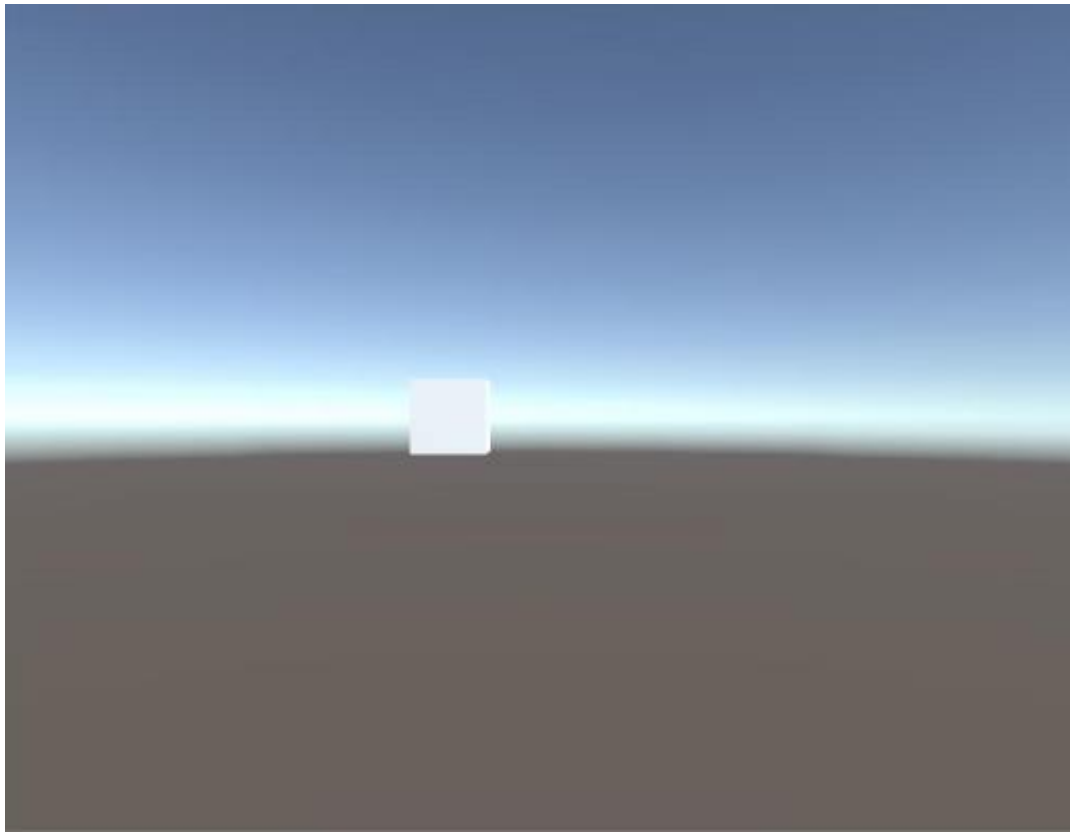
Time 클래스

- 유니티로부터 시간 정보를 얻는 인터페이스

클래스 멤버	내용
captureFramerate	스크린샷을 찍기 위해 프레임간 재생시간을 느리게 한다.
deltaTime	지난 프레임과 현재 프레임 사이의 시간 간격
fixedDeltaTime	물리 엔진의 지난 프레임과 현재 프레임 사이의 시간 간격
fixedTime	게임이 시작한 후 물리 엔진이 작동해온 시간
frameCount	게임이 시작된 후 수행한 프레임 수
time	게임이 시작한 후 지나간 시간

직접 해 봅시다

- 움직임에 속도 적용
- 속도 변수를 에디터에서 변경
  - 클래스 멤버 중 public 권한으로 설정한 멤버 변수는 Inspector View에 나타난다.



RANDOM 클래스

Random 클래스

- 난수 데이터를 생성하는 클래스

클래스 멤버	내용
insideUnitCircle	범위가 1인 원 안의 한 점을 무작위로 생성
insideUnitSphere	범위가 1인 구 안의 한 점을 무작위로 생성
onUnitSphere	범위가 1인 구 표면의 한 점을 무작위로 생성
rotation	회전 값을 무작위로 생성
rotationUniform	균등 분포 내에서 회전 값을 무작위로 생성
seed	난수 발생기에 씨드 넘버를 설정
value	0.0~1.0 사이에 값을 무작위로 생성
Range(float min, float max);	설정한 범위 내의 값을 무작위로 생성

유니티 스크립팅 레퍼런스

- 주요 클래스의 내용 확인