

# CIEN Unity 3D

중급반 (화요일 오후 7시)

고주형



# 목차

1. 스크립트 개념
2. 입력 처리
  - 입력 받아서 이동해보기
3. Frame과 deltaTime
4. 이동과 회전
5. 에셋스토어

실습. Flappy Bird




# 1. 유니티 스크립트

Frame, Start, Update

# 스크립트

- Script
  - 우리가 만드는 기능
  - 설계 도면
  - 아직 실체화는 X
    - Quiz. Class와 Instance의 차이점
- 유니티의 컴포넌트만으로는 우리가 원하는 게임을 구현하기 어렵다
  - 원하는 게임 = 유니티 컴포넌트 + 스크립트

# 스크립트를 만들어보자



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

public class NewBehaviourScript : MonoBehaviour
{
    public int ThisIsPublic = 1;

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

    }

    // Update is called once per frame
    void Update()
    {

    }
}
```

처음 생성했을 때의  
코드

# 스크립트를 만들면

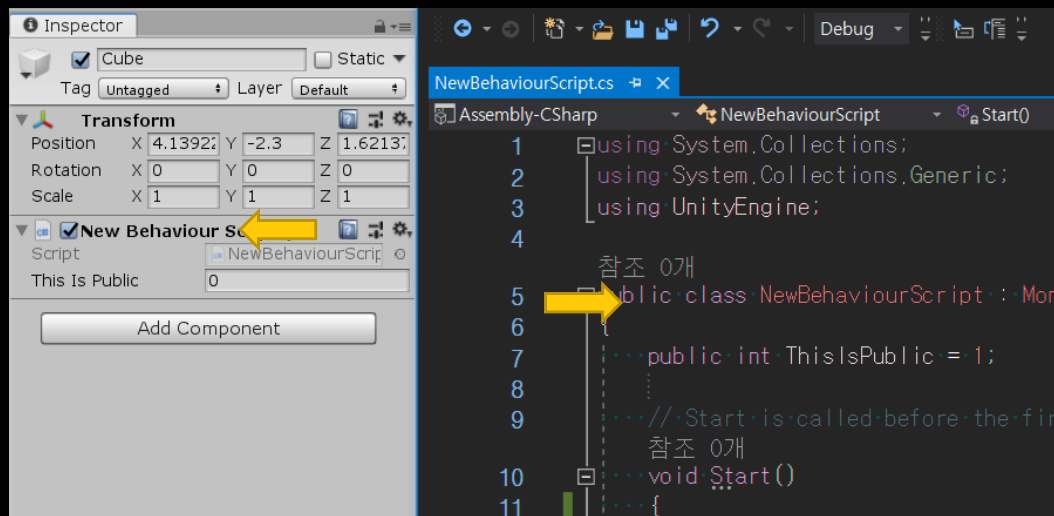
- **Start()**
  - 게임이 시작할 때 **한 번만** 호출된다
  - Update 호출 전에 호출!
  - 주로 변수를 초기화할 때
  - 생성자와 비슷한 느낌 (개인적인 의견)
- **Update()**
  - **프레임마다** 호출된다
  - Input을 받은 후 호출된다
- 유니티가 자동으로 호출해주는 특별한 함수

# 특별한 스크립트의 이벤트 함수들

- 유니티가 자동으로 호출해주는 특별한 함수
- **MonoBehaviour**로부터 상속 받음
- 이런 이벤트 함수들의 실행 순서:  
<https://docs.unity3d.com/Manual/ExecutionOrder.html>

# 스크립트를 컴포넌트로 추가할 때

- **public**인 변수들은 inspector 창에서 조절할 수 있다
- **Inspector** 창에 노출되는 순간부터 따로 존재하게 됨
  - 초기값으로 안 돌아옴
  - 다른 메모리로 존재
  - 실제로 쓰이게 되는 값
  - **Reset**(기본값 모든 값을 원위치!)





# 메시지 출력하기

```
Debug.Log("메시지");
```

- Console창에서 메시지 출력하기
- 제대로 작동하고 있는지 확인용
- 오류 출력할 때



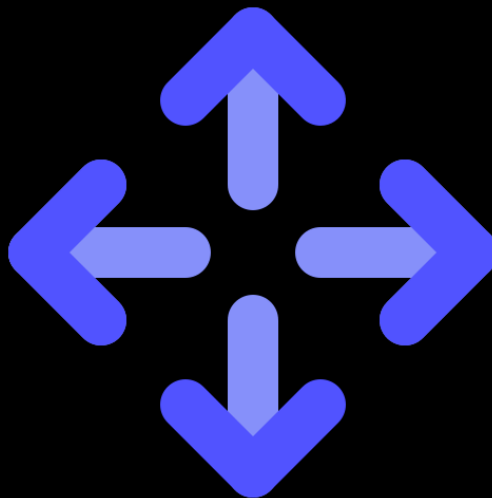
```
Debug.Log("Test 메시지");  
Debug.Log("Player 사망");
```

# Document

다음의 내용을 찾아보자!

컴포넌트로 만들 때 항상 상속 받는 `MonoBehaviour`

- `Start`
- `Update`
- `FixedUpdate`
- ...



## 2. 입력 처리

`Input.GetKey(KeyCode)`

`Input.GetKeyDown(KeyCode)`

`Input.GetKeyUp(KeyCode)`

# 입력

## Input Class

- 키보드
- 마우스
- 조이스틱
- 스마트폰 터치

# 키보드 입력 받기

## 키보드

- **Input.GetKey**(키-코드)      // 눌린 상태
- **Input.GetKeyDown**(키-코드)    // 누를 때
- **Input.GetKeyUp**(키-코드)      // 떴을 때

## 마우스 클릭

- **Input.GetMouseButton**(숫자) // L(0), R(1), Wheel(2)
- **Input.GetMouseButtonUp**(숫자)
- **Input.GetMouseButtonDown**(숫자)

## 마우스 위치

- **Input.mousePosition**

# Input 테스트



```
Update()  
{  
    Debug.Log("W 누름? " + Input.GetKey(KeyCode.W));  
    Debug.Log("A 누름? " + Input.GetKey(KeyCode.A));  
    Debug.Log("S 누름? " + Input.GetKey(KeyCode.S));  
    Debug.Log("D 누름? " + Input.GetKey(KeyCode.D));  
}
```

# Vector

대부분의 (물리) 값들은 Vector3로 표현되어 있다

(ex) 위치, 회전, 크기

Vector3? x, y, z 값을 가진 구조체(struct)

- x, y, z는 float형임을 주의하자!
- Struct(구조체)이다!
  - Quiz. Class와 Struct의 차이점?
- 벡터 연산 가능하다!
  - 외적/내적
  - 벡터 사칙연산

(ex)

```
transform.position = new Vector3(0, 0, 0);
```

# Vector3 혼한 실수

- 1. Type Casting

```
// Start is called before the first frame update
참조 0개
void Start()
{
    Vector3 vector = new Vector3(1.0, 1.0, 1.0);
}

// Update is called once per frame
```

struct System.Double  
3 인수: 'double'에서 'float'(으)로 변환할 수 없습니다.

- 2. Value Type vs Reference Type

```
public class NewBehaviourScript : MonoBehaviour
{
    Vector3 myPosition;

    // Start is called before the first frame update
    참조 0개
    void Start()
    {
        myPosition = transform.position;
        myPosition.x = 10;
        myPosition.y = 10;
        myPosition.z = 10;
    }
}
```



# Copy of Struct's Member

- 1. 이걸 왜 안될까요?
- Property라는 문법 때문에 오류가 잘 안보인다.

```
10  □ 참조 0개  
11  │ void Start()  
12  │ {  
13  │ │ transform.position.x = 10;  
14  │ │ transform.position.y = 10;  
15  │ │ transform.position.z = 10;  
16  │ }
```

- 실제로 => `transform.GetPosition().x = 10`
- 만약 Class였다면? ㄱㅈ

# Good



```
Update()
{
    transform.position = new Vector3(transform.position.x + 0.1f, transform.position.y, transform.position.z);
}
```

# Better

```
// 월드 기준  
transform.Translate(new Vector3(10, 0, 0), Space.World);  
// 로컬 기준  
transform.Translate(new Vector3(10, 0, 0), Space.Self);
```

# 입력을 받아서 이동해보자

- 플레이어 이동 구현
- 키(wasd) 누름 -> 위치 변경



# 3. Frame과 deltaTime

Frame Per Second(FPS)

Time.deltaTime

# Frame

게임은 무한 루프

- 매 루프 => 한 프레임 = 한 장면

Frame Per Second(FPS)

- 60 FPS => 대충 1초에 60프레임
- 1프레임의 시간은 그때 그때 컴퓨터의 상황에 따라 다름

# 성능 무관 - $\Delta time$ 이용

- `Time.deltaTime`
- 이전의 프레임과 현재 프레임 사이의 간격 시간
- 기기마다 성능이 다르다.
  - 1초에 1프레임
  - 1초에 2000프레임
- 시간은 누구에게나 평등
  - 모든 변화(크기, 이동, ...)를 시간을 관여시켜라



## 4. 이동과 회전

Position, Velocity, Force, Rotate

Local, Global



# 위치

Local 부모-나 사이의 위치

- `transform.localPosition`

Global 월드 기준의 절대적인 위치

- `transform.position`

# 이동하기

## Local

- `transform.Translate(x, y, z, Space.Self);`

## Global

- `transform.Translate(x, y, z, Space.World);`

# 속도

- `position += position + velocity;`

# 가속도

- `velocity += velocity + acceleration;`
- `position += position + velocity;`

# 회전

## Local

- `Rotate(x축, y축, z축, Space.Self);`

## Global

- `Rotate(x축, y축, z축, Space.World);`



## 5. Asset Store

무료/유료 Asset을 구매할 수 있는 곳

사운드, 3D 모델, 효과, 툴, ...

# Asset Store

- 유료/무료 에셋을 구매하고 다운로드하는 곳
  - 툴, 사운드, 효과, 아이콘, ...
- 실습용을 쓸 것을 다운로드해보자

# Flappy Bird 점프 구현하기

- 계속 점프가 가능해야 한다
- 컴퓨터 성능과 무관해야 한다



# 참고 문서

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