고주형

https://github.com/IamGroooooot/cien-unity-2019

2019/11/20


복습

Week 3

알아 두면 편한 것들

싱글턴 패턴
- Singleton?
- Singleton C#
- Singleton Unity

코루틴(+Invoke)
- Coroutine을 이해해보자
- Coroutine을 왜 쓸까요?
- Invoke
- Use Coroutines

세이브 파일
- PlayerPref
- Save and Load

사운드
- 하나의 오디오
- 여러 개의 오디오

빌드
- 여러 기기 빌드

싱글턴 패턴
(Singleton Pattern)

Singleton Pattern

언제 쓰기 적합할까요?

- Class가 하나의 인스턴스(오브젝트)만 있도록 하기위해서 씀
- 병행적으로 접근될 공유 자원(값)들을 관리
- 여러 군데에서 그 자원(값)에 접근함

남용하면 안 되지만 게임 제작할 때

- Global하게 접근할 수 있기 때문에 사용 많이 함. (편함)

참고)

병행? Concurrency is the composition of independently executing computations

== 순서에 상관없이 동시에 수행될 수 있다

Singleton C#

- 대충 감을 잡기 위해서
- 먼저 저번 학기 C# 기초반 수업을 할 때 쓴 자료를 봅시다

정적 필드 - 싱글턴(singleton)

특정 클래스의 인스턴스를 의도적으로 딱 1개만 만들고 싶을 때 쓰인다.

- 클래스 내부에 미리 인스턴스를 생성
- 생성자를 private접근 제한자로 명시하여 외부에서 생성하지 못하게 막는다.

정적 필드 - 싱글턴(singleton)

싱글턴 정의

```
class Star
{
        // 클래스 내에서 인스턴스 미리 생성
        static public Star Earth = new Star("지구");

        string Name;

        // private으로 외부에서 객체가 생성되는 것을 막음
        private Star(string name)
        {
                Name = name;
        }

        // public 인스턴스 메서드
        public void DisplayStarName()
        {
                Console.WriteLine(Name);
        }
}
```

정적 필드 – 싱글턴(singleton)

싱글턴 사용

```
static void Main(string[] args)
{
        // 정적 필드로 하나만 존재하는 인스턴스에 접근
        Star.Earth.DisplayStarName();

        // 오류
        // 생성자가 private이므로 외부에서 객체를 생성할 수 없음
        Star Sun2 = new Star("다른 태양");
}
```

Unity의 Singleton

```
public class SomeClass : MonoBehaviour {
    private static SomeClass _instance;

    // get만 할 수 있게
    public static SomeClass Instance { get { return _instance; } }

    private void Awake()
    {
        // 만약에 instance가 이미 존재하면 새로운 인스턴스를 삭제 - Scene 여러 개 만들 때
        if (_instance != null && _instance != this)
        {
            Destroy(this.gameObject);
        } else {
            _instance = this;
        }
    }
}
```

코루틴(Coroutine) 개념

코루틴을 이해해보자

제가 TV로 유튜브를 보기 시작했지만 광고가 나옵니다. 광고를 보는 대신 롤을 실행해서 큐를 돌립니다(대전 상대를 찾는 것).

하지만 롤은 10명이 모여야 시작돼서 많이 기다려야 합니다. 이걸 기다리는 대신 문제 풀이 과제를 합니다.

첫 번째 문제를 풀고 두 번째 문제를 봤는데 Netflix 영상을 보고 풀라고 합니다. 그래서 Netflix를 틉니다. Netflix영상의 Intro을 보는 대신 광고가 끝난 유튜브를 봅니다. 그런데 광고가 다시 뜹니다(2회 광고 ㅂㄷㅂㄷ).

그거를 보던 중 롤 큐가 잡혀서 롤을 합니다 …

그래서 코루틴은 대충 이런 것

- 모든 것을 동시에 일어나게 하는 것처럼 빠르게 작업(함수)들을 전환하는 것이 아니다
- 필요로 할 때, 다른 일을 하는 것

예)

- 필요로 할 때: Space바를 눌러서 3초마다 총을 발사해야 됨
- 일: 총알을 3초마다 계속 쏜다.

코루틴

- Thread와 비슷 but 호출한 곳으로 돌아갈 수 있음.
- 즉, 자기 상태를 기억 함.
- Thread와 똑같은 문제 겪음. Ex) 움직여(우로) & 움직여(좌로)
- 매 프레임마다 체크하여 돌아갈 지점 정함

- Coroutine 실행: StartCoroutine

코루틴을 왜 쓸까요?

- 매번(Update에서 실행하는 것이 아니라) 원할 때만 실행할 수 있음 => 성능 향상
- 가독성 향상
- Update() 함수에 종속적이지 않음.
- 시간 기반의 서브루틴(함수)을 만들기 쉬움.

(Ex) 총알 5초마다 발사

- Update=>
  매 프레임마다 Timer실행 + if문으로 실행할지 말지 체크
- Corountine=>
  Yield return new waitforsecond(5f);

(참고용)
Coroutine vs Invoke

Invoke

- 코루틴이랑 똑같아 보이는 invoke는 뭘까
- 원하는 시점에서 함수 호출하고 싶을 때
- InvokeRepeating으로 함수를 반복적으로 사용 가능
- 코루틴과의 차이
  - 내부적인 타이머로 작동함
  - (중요) 인자를 전달할 수 없다

Invoke를 사용해보자

- InvokeRepeating("함수 명", 몇초 후 시작할지, 몇초마다 반복할지);

```
void Start(){
        InvokeRepeating(“DebugSomething”, 1f, 2f);
}

void DebugSomething(){
        Debug.Log(“1초 지연 후, 2초마다 반복하기”);
}
```

코루틴(Coroutine)을 써보자

코루틴 말들기

- 코루틴 만들기 (IEnumerator):
  - IEnumerator(열거자)와 yield 키워드를 사용해서 만든다

- 특정 코루틴 실행 (StartCoroutine):
  - StartCoroutine("코루틴함수명",인자);
  - StartCoroutine(코루틴함수명(인자)); // 굳

코루틴 주의

1. Coroutine함수는 일반 함수처럼 호출하면 실행되지 않는다.
   그런데 아무런 경고도 뜨지않는다!!
   반드시 StartCoroutine을 사용해서 호출해야 한다.

2. Coroutine은 WaitForSecond는 Time.timeScale의 영향을 받는다.
   Time.timeScale이 x이면; WaitForSecond(1) => 1초/x 대기
   - Time.timeScale이 1일 때; WaitForSeconds(1) => 1초 대기
   - Time.timeScale이 0.5일 때; WaitForSeconds(1) => 2초 대기

3. StartCoroutine을 호출하는 스크립트가 붙은 오브젝트가 비활성화 되거나
   파괴되면 Coroutine은 중단된다.

지원하는 Yield문들 중 일부입니다.

Yield하는 Data에 따라 유니티가 다르게 작동합니다.

- yield return null: 다음 프레임까지 대기
- yield return new WaitForSeconds(float): 지정된 초 만큼 대기
- yield return new WaitForFixedUpdate(): 다음 물리 프레임까지 대기
- yield return new WaitForEndOfFrame(): 모든 렌더링 작업이 끝날 때까지 대기
- yield return StartCoRoutine(string): 다른 코루틴이 끝날 때까지 대기
- yield return new WWW(string): 웹 통신 작업이 끝날 때까지 대기
- yield return new AsyncOperation: 비동기 작업이 끝날 때까지 대기 (씬로딩)
- Yield reutn new WaitUntill(()=> hp<0);

간단한 코딩

```
// 코루틴 실행
void Start()
{
    StartCoroutine(RepeatDebug());
    // StartCoroutine(“RepeatDebug”);도 가능!
}

// 1초마다 디버그를 찍어보자
private IEnumerator RepeatDebug()
{
    while (true)
    {
        yield return new WaitForSeconds(1f);
        Debug.Log("나는 그루트다");
    }
}
```

WaitForSecond 재사용

1초마다 반복되는 new가 불-편하다.
(new연산은 무겁다)

WaitForSecond를 재사용해보자

간단한 코딩

```
// 코루틴 실행
void Start()
{
    StartCoroutine(RepeatDebug());
}

private IEnumerator RepeatDebug()
{
    WaitForSeconds sleepOneSec = new WaitForSeconds(1f);

    while (true)
    {
        yield return sleepOneSec;
        Debug.Log("나는 그루트다");
    }
}
```

유니티로 저장하고 불러오기
(Save File)

Save File

- 유니티의 기능 쓰면 빠르게 만들 수 있음
- PlayerPref 쓰면 편하다

- Save: **SetString**(string **key**, string **value**);

```
>> playerName = PlayerPrefs.SetString("Name“, ”고주형”);
```

- Load: **GetString**(string **key**);

```
>> playerName = PlayerPrefs.GetString("Name“);
```

PlayerPref의 Static Method들

DeleteAll	Removes all keys and values from the preferences. Use with caution.
DeleteKey	Removes key and its corresponding value from the preferences.
GetFloat	Returns the value corresponding to key in the preference file if it exists.
GetInt	Returns the value corresponding to key in the preference file if it exists.
GetString	Returns the value corresponding to key in the preference file if it exists.
HasKey	Returns true if key exists in the preferences.
Save	Writes all modified preferences to disk.
SetFloat	Sets the value of the preference identified by key.
SetInt	Sets the value of the preference identified by key.
SetString	Sets the value of the preference identified by key.

PlayerPref이 싫으면?

- 파일 입출력!
  - Json, CSV, XML, … 원하는 format으로 작업 가능

```
{
    "이름": "고주형",
    "학번": 18,
    "성별": "남",
    "주소": "흑석동 CIEN",
    "특기": ["코그모", "코딩노예"],
    "졸업예정자": false
},
{
    "이름": "그루트",
    "학번": 18,
    "성별": "남",
    "주소": "아스가르드",
    "특기": ["I", "Am", "Groot"],
    "졸업예정자": true
},
...
```

```
생일,    이름,   학번, 성별,  특기
981003, 고주형,  18,  "남",  "코그모, 코당노예"
000101, 그루트,  99,  "남",  "I, Am, Groot"
```

사운드 넣기
(Audio)

Sound

- Audio-Listener (듣는 곳)
- Audio-Source (사운드)
  - 하나의 Audio Clip 재생: Play, Pause, Stop
    - **Play**(ulong **delay** = 0)
      ```
      >> audioSource.Play(0); // 바로 재생
      ```
    - `>> audioData.Pause(); // 일시정지`
    - `>> audioData.UnPause(); // 다시 재생`
    - `>> audioData.Stop(); // 정지`

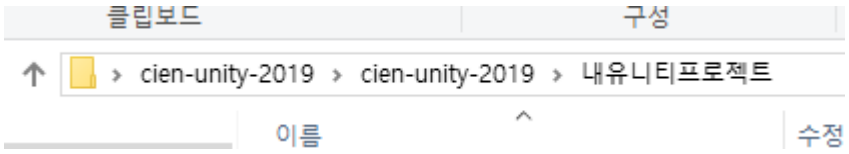
  - 2개 이상 Audio Clip 재생:

```
>> AudioClip myAudio;
>> float volumeScale = 0.5f; // 값 범위 0~1
>> audioSource.PlayOneShot(myAudio, volumeScale);
```

Build

- 앱(안드) = Android Studio – SDK 필요한 버전들 다 다운 빌드
- 앱(iOS) = 맥 필요, Mac이 아니라서 모릅니다 ㅠ
- HTML5: 다운받고 빌드
- PC: 바로 빌드 가능

- 주의) 유니티 프로젝트의 경로에 한국어(아스키가 아닌 것)가 있으면 오류가 날 수 있습니다.



공지

UI 특강 수요 조사

UI 특강 (UI 장인 김수환)

- 이번주 금요일 11/22
- 오후 (저녁쯤에)할 예정

- 되시는 분!!
- 시간이 된다면 (1명) + 튜프빨리끝나면(2명)

출처

코루틴

- quote from Idan Arye
- https://linecode.tistory.com/9 [개발자도 한줄코딩부터..!]
- https://12bme.tistory.com/184 [길은 가면, 뒤에 있다.]
- https://teddy.tistory.com/22 [Teddy Games]
- http://theeye.pe.kr/archives/2725 <- 유니티 코루틴에 대해 더 공부하고 싶으면 참고하세요!

PlayerPref

https://docs.unity3d.com/ScriptReference/PlayerPrefs.html