유니티에서 싱글톤 패턴 사용하기

게임을 만들다 보면 각 오브젝트 및 데이터를 관리하는 매니져 클래스 / 오브젝트를 만들어서 관리하게 됩니다.

이렇게 관리되는 데이터들은 다른 오브젝트에서도 접근이 가능해야 하는 경우가 많은데, 컴포넌트 기반 엔진인 유니티에서는 접근이 용이하지 만은 않습니다.

함수들은 SendMessage를 이용해서 접근이 가능하지만, 변수 등에 접근하려고 할 때는 곤란한 경우가 많습니다.

이럴 때 사용하면 좋은 패턴이 Singleton(싱글톤) 패턴입니다.

다른 오브젝트도 매니져 클래스에서 public으로 선언된 것에 대해서는 직접 접근이 가능합니다. 변수, 함수 모두 가능합니다.

예제 코드를 보며 확인해보겠습니다.

Singleton을 구현하는 방법은 정말 많습니다. C#에서만 사용되는 기술이 아니라서 다른언어에서도 많이 사용되고 있는 보편적인 패턴이라고 할 수 있습니다.

아래 예제 코드는 그 중에서도 제가 주로 사용하는 코드 형태입니다.

|  |
| --- |
| using *UnityEngine*;  using *System*.*Collections*;    public class ManagerClass : *MonoBehaviour*  {  private static ManagerClass \_instance = null;  public static ManagerClass Instance  {  get  {  if (\_instance == null)  {  \_instance = *FindObjectOfType*(typeof(ManagerClass)) as ManagerClass;  if (\_instance == null)  {  *Debug*.*LogError*("There's no active ManagerClass object");  }  }  return \_instance;  }  }  } |

코드를 살펴보면, 먼저 자기 자신 type의 (여기서는 ManagerClass 타입) private 형 static 변수를 갖습니다.

그리고 외부에서 접근할 수 있도록 프로퍼티를 public으로 선언하고 접근할 때 \_instance가 null이면 FindObjectOfType 함수를 이용해서 씬 내에 있는 ManagerClass 오브젝트를 찾습니다.

검색을 해보고 검색에 성공한 경우 그 값을 반환하고, 검색에 실패하면 오류를 발생하도록 구현을 했습니다.

FindObjectOfType 함수는 유니티에서 비용이 큰 함수 중 하나지만, 잘 살펴보시면 아시겠지만, 처음 한번만 호출됩니다.

이제 ManagerClass에 아래와 같이 간단한 함수를 public으로 선언을 하고 접근을 해보겠습니다.

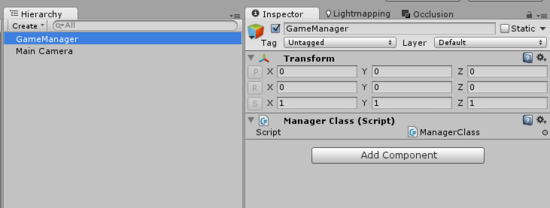
|  |  |
| --- | --- |
| 1  2  3  4 | public void TestFunc()  {      Debug.Log("TestFunc() Called");  } |

외부에서 접근을 할 때는 ManagerClass.Instance.TestFunc()와 같은 방식으로 합니다.

이렇게 하면 SendMessage를 사용하지 않고 바로 함수에 접근이 가능합니다. 변수도 마찬가지 입니다. ManagerClass.Instance.변수이름 이렇게 접근하면, public으로 선언된 변수, 프로퍼티는 모두 접근할 수 있습니다.

테스트를 해보겠습니다.

먼저 ManagerClass 스크립트를 갖는 게임 오브젝트를 하나 생성합니다. 게임 오브젝트 이름은 GameManager 라고 지정했습니다.

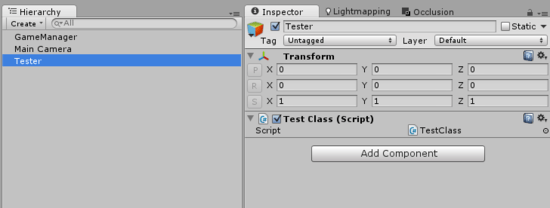


그리고 ManagerClass의 TextFunc() 함수에 접근하는 클래스를 생성합니다.

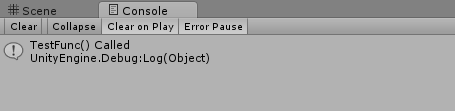
아래 스크립트처럼 간단히 Start 함수에서 호출하도록 했습니다.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | using UnityEngine;  using System.Collections;    public class TestClass : MonoBehaviour  {      void Start()      {          ManagerClass.Instance.TestFunc();      }  } |

그런뒤 아래 그림처럼 이 스크립트를 갖는 게임 오브젝트도 생성합니다.



이제 실행을 하고 결과를 살펴보겠습니다.



위에 보는 것과 같이 호출이 잘 되는 것을 확인할 수 있습니다.

게임을 만들다가 매니져(관리자) 클래스가 필요한 경우 Singleton 패턴을 사용하면 이처럼 편리하게 접근이 가능합니다.

내용 끝까지 읽어주셔서 감사합니다.

배너 클릭은 저에게 많은 힘이 됩니다.