게임프로젝트 참고서

수정 2018.05.30

1. 게임 디자인 패턴
2. 싱글 톤

흔히 Manager 클래스를 제작하고, 단일 매니저를 활용한 구조를 구현할 때 사용한다.

유니티 내 씬 이동시 모든 오브젝트가 삭제되므로,. 꼭 이 패턴을 사용할 때는

DontDestoryOnLoad( this.gameObject );’ 를 사용하여 파괴를 막아야 한다.

선언 방법은 클래스 내부에 public static <선언한 클래스이름> instance 로 instance 변수를 선언 하여 클래스이름.instance 로 싱글 톤에 접근한다. 하지만 매번 이런 instance 선언을 하는 불편을 해소 하기 위해 제너릭 싱글톤을 구현하고 선언하는 방법도 있다.

링크 <http://ronniej.sfuh.tk/unity3d-%EC%A0%9C%EB%84%A4%EB%A6%ADgeneric%EC%9D%84-%EC%A0%81%EC%9A%A9%ED%95%9C-%EC%8B%B1%EA%B8%80%ED%86%A4-%EB%A7%A4%EB%8B%88%EC%A0%B8-%EB%A7%8C%EB%93%A4%EA%B8%B0/>

1. FSM (Finite State Machine)

흔히 상태 패턴 이라고 불리며, 매 프레임 마다 현재 상태를 확인하여 처리하고, 특정 이벤트가 발생 하여 상태를 변경 시키는 패턴이다. 가깝게 볼 수 있는 곳은 유니티 애니메이터 ( Animator ) 의 상태 창이다. 패턴의 핵심은 Update 함수처럼 코루틴이 무한 루프로 돌지만, 프로그래머가 원하는 타임 또는 타이밍에 상태를 체크할 수 있다. 또한 상태를 Enum으로 선언하여 직관적으로 구현하는 것도 팁이다.

1. Observer

관잘차 패턴이라고 불리며, 간단하게 생각하면 , 알림, 업적 같은 기능이다.

다른 객체에서 요청한 함수들을 하나의 함수에서 관리하여 특정 행동이나 조건이 만족되면 하나의 함수를 실행한다. 특히 Delegate를 이해하고 이 패턴을 구현한다면 편하다. Delegate 함수에 작동되야 하는 함수를 모두 넣고 특정 조건에 Delegate를 실행 하면 그것이 Observer 패턴이다.

1. 컴포넌트

유니티에서 기본적으로 지원하는 패턴이다. 각 클래스가 독립적으로 움직이고, 연동이 필요한 클래스들은 내부에서 해당 컴포넌트를 찾아 연결한다.

참고 사이트 - http://rapapa.net/?p=3198

1. 최적화 기법
2. 컴포넌트 캐싱

최적화의 기본이 되는 기법이다. 매번 사용하는 GetComponen() 함수는 Awake나 Start 함수에서 1번만 사용하여 변수로 저장해두고 사용한다.

1. Instantiate / Destory 사용

Instantiate (오브젝트 생성) 와 Destory (오브젝트 파괴) 는 렉의 주 원인이다. C# 언어 특성상 GC(가비지콜렉터) 가 구현되어있는데 이 GC는 언제 실행되고 사라질지 모르는 것이기 때문에 GC 실행에 많은 원인이 되는 두 함수는 사용을 제한 하는게 좋다.

1. 오브젝트 풀링 ( Object Pool )

GC 실행의 문제를 해결하기 위해 구현되는 기법이 오브젝트 풀링이다. 슈팅게임과 같이 많은 총알이 생성되어야 할 때, 게임을 시작하기전에 미리 화면에 표시되는 최대 개수 만큼 생성하여 생성된 오브젝트만을 가지고 재활용하는 기법이다. 이로 인해 GC가 적게 돌 가능성이 생긴다.

1. 박싱 / 언 박싱

프로그래밍 작업시 흔히 작성하게 되는 (int) a, (float) a, data.ToString() 과 같이 자료형의 변환에서 많은 연산이 필요하다. 되도록이면 공통적인 자료형이나 클래스를 사용하는 것이 좋다. 박싱/언박싱의 부분 최적화는 많은 생각과 토론이 필요한 부분이다.

1. 게임 오브젝트 태그의 비교

종종 물리 처리, 오브젝트 확인을 위해 사용하는 if( gameObject.tag = “태그이름” ) 보다는 if ( gameObejct.CompareTag(“태그이름”) ) 이 훨씬 빠르다. 이유는 gameObejct.tag 를 실행하면 태그 자체를 메모리에 할당 해야되기 때문에.

1. “==” vs “equals()”

데이터 비교를 위해 사용되는 == 연산자는 임시적인 메모리가 남는다. 되도록 equals() 를 사용하는게 좋다.

1. “class” vs “struct”

Class 는 heap 구조로 처리되어 GC에 들어가게 된다. 하지만 복사되지 않기 때문에 메모리는 그대로 남아 절약 할 수 있다.

Struct 는 Stack 구조로 GC에 들어가지 않는다. 하지만 모든 데이터가 통째로 복사되어 메모리로 들어가기 때문에 데이터가 큰 구조체를 사용하면 안된다. 구조체 내부에 3~4개 의 변수 정도가 적당하다. 10개 이상은 클래스로 변환하자.

1. 라이프 사이클에 있는 콜백 함수들

유니티에서 기본적으로 선언되어 작동하는 함수들이 몇가지 있다.

Reset > Awake > OnEnable > Start > FixedUpdate > 충돌 처리 > Update > LateUdpate 순으로 실행되며, OnDisable, OnDestory 도 존재한다.

이런 함수는 선언만 되어있으면 매 프레임 마다 실행된다. 함수 내부가 비어있어도 실행되기 때문에 사용하지 않으면 과감히 없애야 한다.

1. FindObject 사용

검색 비용이 꽤나 많이 나가는 함수이기 때문에 되도록 초기 설정 부분에서 검색해야한다.

1. Update vs Coroutine

매번 실행해야 한다면 매 프레임마다 실행되는 Update도 좋지만, 한번만 여러번 진행되야 하는 함수인 경우 Coroutine으로 작업을 실행하고 없애버리는 것도 좋은 최적화 기법이다.

1. 1 / 10 vs 1 \* 0.1

나눗셈은 연산이 많이 느리다. 극히 일부분을 제외하곤, 곱셈으로 구현하는걸 추천한다.

1. 오브젝트의 거리 비교

Distance함수나 magnitude 보다는 sqrMagnitude를 무조건 사용한다. 제곱근 연산으로 루트를 푸는 연산이 매우 오래 걸린다. 단 sqrMagnitude는 루트 값이 적용되있지 않으므로 값에 신경써야 한다.

1. 프로그래밍 기법
2. 전역 변수와 지역변수 이름 규칙

보통은 다음과 같은 규칙을 사용한다.

전역 변수 : value , 지역 변수 : \_value

물론 코드가 자신이 작성하고 싶은 마음대로 작성되지만, 해당 코드를 읽는 사람도 있다는 점을 명심하자.

1. 네이밍 규칙 ( 파스칼 VS 카멜 )

파스칼 : HelloWorld

카멜 : helloWord

1. 인터페이스

항상 인터페이스 클래스는 I클래스이름 으로 앞에 I 를 붙어야한다.

1. 축약어 금지

흔히 사용하는 i, j , k 같이 루프문에 필요한 변수를 제외하고는 풀네임을 적는 것이 좋다. 물론 서로 약속한 축약문은 사용해도 좋다.

Ex ) transform > tr / rigidbody > ri or rb (별로 추천하지는 않음)

1. 주석이나 서머너리

주석은 // 주석 내용 으로 주석을 작성하고, 서머너리는 /// 설명내용 으로 작성하지만, 주석이나 서머너리가 필요 없는 네이밍이 최고로 좋다. 되도록 해당 이름이 다른 프로그래머에게 인식되기 쉬운 네이밍으로 하는 것이 포인트.

1. Boolean 타입 네이밍

이름 앞에 is 를 붙여서 isFind 같이 사용하는 것이 좋다.

1. 코드를 그룹화 하여 보기

#region

<코드 내용>

#endregion

을 사용하면 + - 버튼으로 코드를 줄이거나 표시 할 수 있다.

1. 유니티 에디터 팁
2. 인스펙터 2개 이상 사용하기

* 인스펙터 위의 자물쇠 이미지를 눌러서 잠군후 AddTap으로 인스펙터를 하나 더 만들어 사용하면 편리하다.

1. 플레이 모드 구별하기

* Edit >Preference 의 Colortint 를 수정하면 구별하기 편리하다.

1. 피봇이나 모서리로 이동하기

* V 키를 누르고 이동하면 편리하다.

1. 포커스 따라가기

* 원하는 오브젝트를 선택하고 마우스를 씬위에 올린 상태에 F키를 누르면 포커스가 움직인다. 이때 한번더 F를 누르면 포커스가 고정되어 이동중인 오브젝트를 자동으로 따라간다.

1. 리소스 구하기 팁

모든 게임 리소스 총괄

https://opengameart.org/

https://assetstore.unity.com/

출시된 게임 리소스

https://www.spriters-resource.com/

그래픽 리소스

http://www.glitchthegame.com/public-domain-game-art/

게임 사운드

https://sonniss.com/gameaudiogdc18 <30GB + >

http://www.glitchthegame.com/downloads/

TCG 일러스트 리소스

http://www.jewel-s.jp/download/

저작권 명시 해야함. タイトル：[JewelSaviorFREE] URL：[http://www.jewel-s.jp/]

개발 예제 검색 [github]

https://unitylist.com/

1. 기타..