유니티에서 병렬처리를 위한 방법

**병렬처리 사용방법**

using System.Threading.Tasks;

Parallel.ForEach(files, (file) => Testtt(file));

Parallel.for 등이 있음

**테스트 함수**

public void Testtt(string tt)

{

Debug.Log(tt);

for (int index = 0; index < 10000; index++)

{

for (int index2 = 0; index2 < 10000; index2++)

{

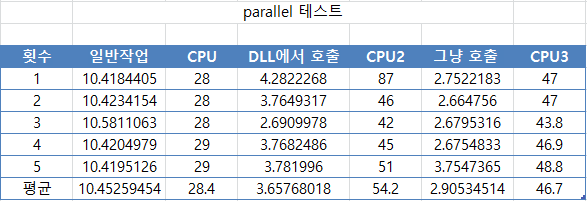
//busy work

}

}

}

**테스트 결과**



오래걸리는 작업을 Parallel을 통해 병렬처리로 전환하여 사용할 수 있다. Task, Thread와 마찬가지로 유니티메인 쓰레드에서 동작하는 기능은 추가 했을 시 에러를 출력한다.

CPU 사용률이 증가하고 동기방식으로 진행 된다. Wait기능까지 호출된다고 MDNS에서 본듯

DLL에서의 호출은 역시 직접 호출보다 속도가 안나온다는 것을 확인 할 수 있다.

**테스트 사용 전체 코드**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using ParallelDLLTest;

using System.IO;

using System.Threading.Tasks;

public class ParrallelStart : MonoBehaviour {

// Use this for initialization

void Start()

{

string currentpath = Directory.GetCurrentDirectory() + @"\Sample";

Debug.Log(currentpath);

DirectoryInfo info = new DirectoryInfo(currentpath);

DirectoryInfo[] infoarray = info.GetDirectories();

string samplepath = string.Empty;

for (int index = 0; index < infoarray.Length; index++)

{

Debug.Log(infoarray[index].Name);

samplepath = infoarray[0].FullName;

}

string[] files = Directory.GetFiles(samplepath);

TimerChecker.TimeChecker.StartTimer(0, "Start parallel");

//for (int index = 0; index < files.Length; index++)

//{

// Testtt(files[index]);

//}

//ParallelDLLTest.Class1.ParallelForEachFunction(files, Testtt);

Parallel.ForEach(files, (file) => Testtt(file));

TimerChecker.TimeChecker.EndTimer(0);

}

public void Testtt(string tt)

{

Debug.Log(tt);

for (int index = 0; index < 10000; index++)

{

for (int index2 = 0; index2 < 10000; index2++)

{

//busy work

}

}

}

}