**Compute Shader初步学习**

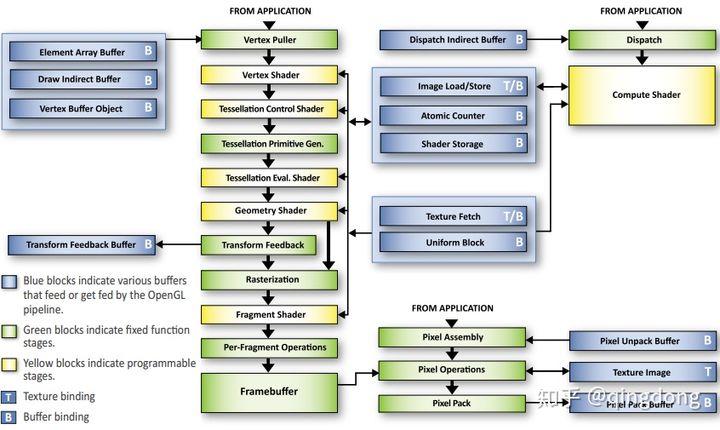
**一、Compute Shader介绍**

1. GPGPU编程概念

GPGPU编程(General Purpose GPU Programming GPU通用计算): CPU和GPU各有自己的擅长，CPU适合做串行处理， 而GPU做并行处理。将二者结合起来的技术就叫做GPGPU，也就是利用GPU进行通用计算的技术。

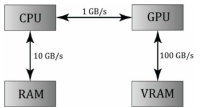
2.Computer Shader概念

Compute Shaders是在GPU运行却又在普通渲染管线之外的程序。用于运行GPGPU program（Opengl4.3的特性（2012））。

OpenGL 4.5 PipleLine

3.适用场景

大量并行少分支的数学运算，任何一个涉及到对数据集合的每一个元素都进行同样一系列操作（无分支）的问题都非常适合它。例如：粒子间互无影响的粒子系统



**二、Unity Compute Shader的使用**

1．总体介绍：

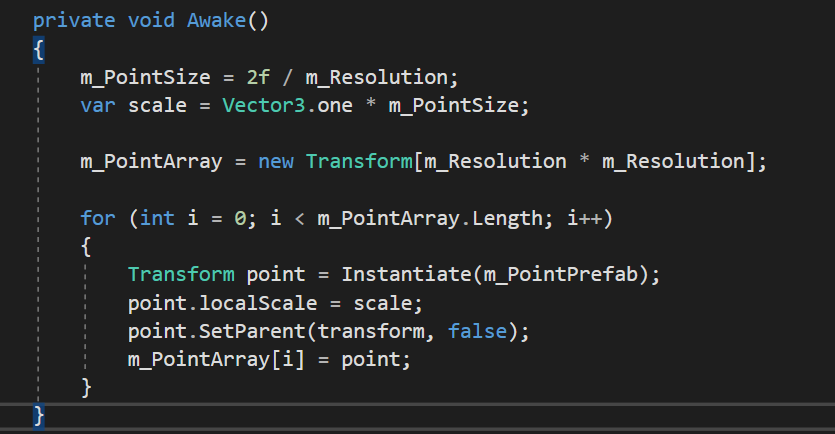
Unity Compute Shader不像普通shader用material渲染物体，它必须由C# Script驱动：

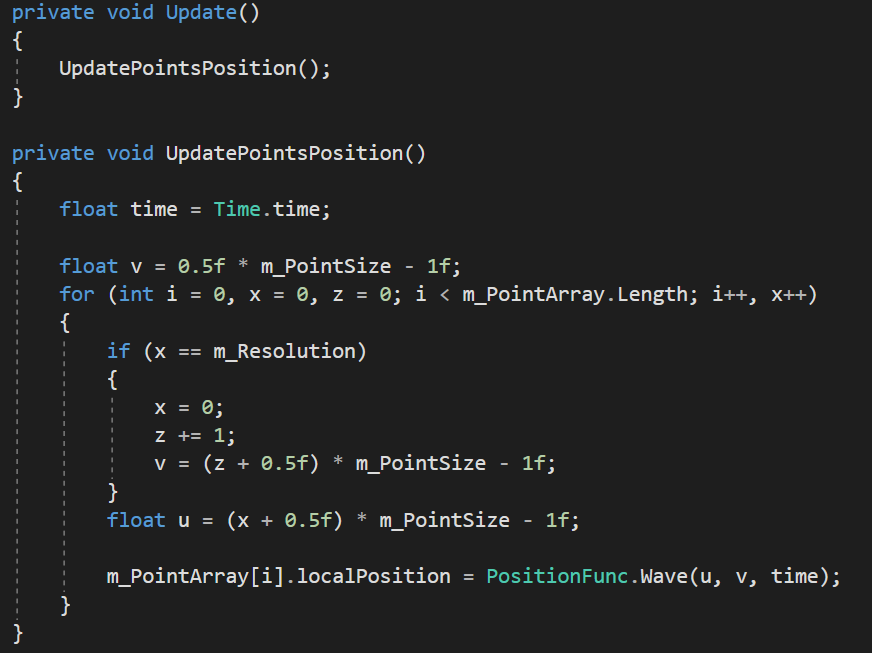
1. C# Script 创建渲染所需的信息包括模型、贴图、参数等然后调用Compute Shader
2. Compute Shader计算出新的模型位置 对贴图进行处理 改变参数等
3. Vertex Fragment Shader获得Compute Shader产生的信息进行渲染

2. 用一个实例讲解Unity Computer Shader的使用

需求：大量物体组成的图形渲染

a.常规解决方案





运行结果：

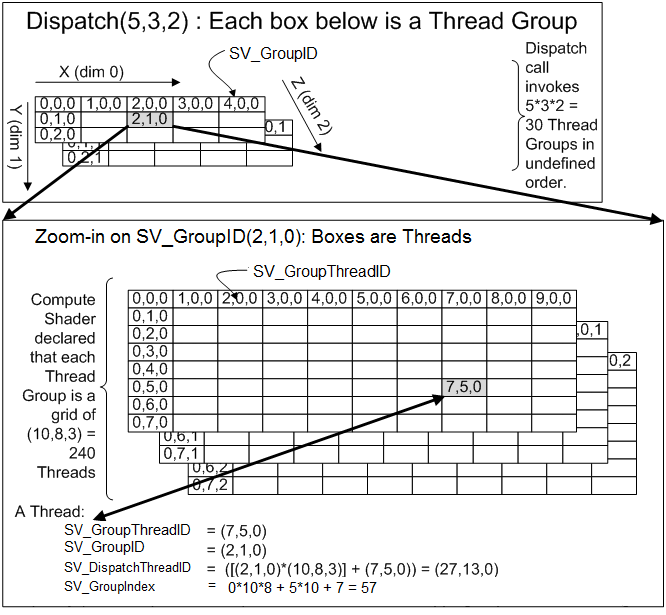
b.Computer Shader方案

**c#部分代码**

**computer shader代码**

**渲染shader代码**

下图是线程和线程组，线程组和分发的总线程组之间的关系。



SV\_GroupThreadID:一个线程组中用来唯一表示这个线程的ID，这个唯一性只在本组内有效，表示形式为三维向量的形式，如上面的在一个线程组中的（7,5,0）

SV\_GroupIndex:一个线程在本线程组中的索引,用整数表示，从0开始，从第一行从左往右累计，然后是第二，三，，，，行，如上面计算出来的57（其实是第58个线程，但对应索引为57，因为起始0占一个）。

SV\_GroupID：一个线程组在整个分发的线程组中的ID，在所有线程组中具有唯一性，表现形式也是三维向量形式，如上面的（2,1,0）。

SV\_DispatchThreadID：是用来表示一个线程在整个分发的全部线程组中的ID,在所有分发的线程中唯一，表示形式为三维向量形式，计算方式：SV\_GroupID x 一个线程组的规格+SV\_GroupThreadID。线程组规格为如上图的（10,8,3）或一致的代码中对应的numthreads(x，y，z)。

https://blog.csdn.net/a359877454/article/details/80275157