# Unity Shader

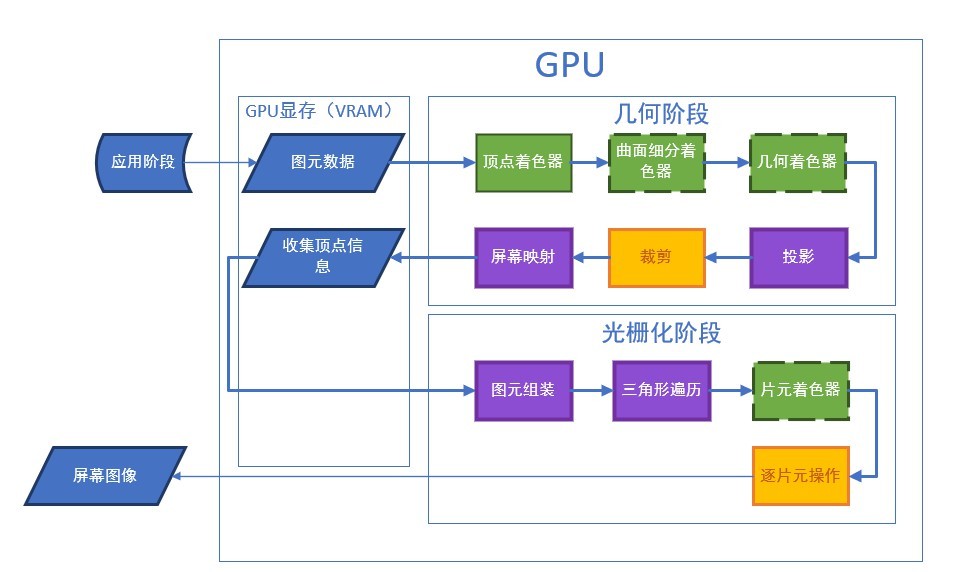
### 什么是shader

Shader,中文翻译即着色器，是一种较为短小的程序片段，用于告诉图形硬件如何计算和输出图形，过去便会

语言来编写，现在也可以使用高级语言来编写。一句话概括：Shader是可编程图形管线的算法片段。 它主要分为两类，Vertex Shader和Fragment Shader

### 什么是渲染管线

渲染管线也称为渲染流水线，是显示芯片内部处理图形信号相互独立的的并行处理单元。一个流水线是一序列可以并行和按照固定顺序进行的阶段。每个阶段都从它的前一阶段接收输入，然后把输入发给随后的阶段。就像一个在同一时间内，不同阶段不同的汽车一起制造的装配线，传统的图形硬件流水线以流水的方式处理大量的顶点、几何图元和片段。



<https://zhuanlan.zhihu.com/p/137780634>

Geometry几何模型——unity——Graphics 引擎调用API——Vertex Processor顶点处理程序——Pixel Processor像素处理器、片段处理器——Frame Buffer帧缓冲

### Shader和材质、贴图的关系

Shader(着色器)实际上就是一种算法，它负责将输入的顶点数据以指定的方式和输入的贴图或者颜色等组合起来，然后输出。绘图单元可以根据这个输出来将图像绘制到屏幕上。输入的贴图或者颜色等，加上对应的Shader，以及对Shader的特定的参数设置，将这些内容（Shader及输入参数）打包存储在一起，得到的就是Material（材质）。之后，我们便可以将材质赋予三维物体来进行渲染（输出）了。

顶点数据 +（这个加号就相当于Shader）贴图or颜色=材质

材质好比引擎最终使用的商品，Shader好比是生产这种商品的加工方法，而贴图or颜色就是原材料。

### 小结 ：

* Shader是图形可编程方案的程序片段
* 渲染管线是一种计算机从数据到最终图形成像的形象描述
* 材质是商品，Shader是方法，贴图/颜色是材料

## Shader三大主流高级编程语言

* HLSL (DirectX)
* GLSL (OpenGL)
* CG (NVIDIA)

Shader language的发展方向是设计出在便捷性方面可以和C++ / Java相比的高级语言，“赋予程序员灵活而方便的编程方式”，并“尽可能的控制渲染过程”同时“利用图形硬件的并行性，提高算法的效率”

Shader language 目前主要有3种语言：居于OpenGL的OpenGl Shading Language，简称GLSL，基于DirectX的High Level Shading Language 简称 HLSL，NVIDIA公司的C for Graphic 简称 Cg 语言。

### OpenGL（GLSL ）介绍

OpenGL (全写Open Graphics Library)是- -个定义了跨编程语言、跨平台的编程接口规格的专业的图形程序接口。它用于三维图像(二维的亦可)，是一个功能强大，调用方便的底层图形库。OpenGL是行业领域中最为广泛接纳的2D/3D图形APL,其自诞生至今已催生了各种计算机平台及设备上的数千优秀应用程序。它独立于视窗操作系统或其它操作系统的，亦是网络透明的。在包含CAD、内容创作、能源、娱乐、游戏开发、制造业、制药业及虚拟现实等行业领域中。OpenGL是个与硬件无关的软件接口，可以在不同的平台如Windows 95、Windows NT、Unix、 Linux 、MacOS、0S/2之 间进行移植。因此，支持OpenGL的软 件具有很好的移植性，可以获得非常广泛的应用。

OpenGL的发展一直处于一 种较为迟缓的态势， 每次版本的提高新增的技术很少，大多只是对其中部分做出修改和完善。1992年7月，SGI公司发布了OpenGL的1.0版本，随后又与微软公司共同开发了WindowsNT版本的OpenGL，从而使--些原来必须在高档图形工作站上运行的大型3D图形处理软件也可以在微机上运用。1995年OpenGL的1.1版本面市， 该版本比1.0的性能有许多提高，并加入了一些新的功能。其中包括改进打印机支持，在增强元文件中包含OpenGL的调用，顶点数组的新特性，提高顶点位置、法线、颜色、色彩指数、纹理坐标、多边形边缘标识的传输速度，引入了新的纹理特性等等。OpenGL1.5又 新增了“ OpenGL Shading Language"，该语言是“OpenGL 2.0”的底核，用于着色对象、顶点着色以及片断着色技术的扩展功能。

### DirectX（HLSL）介绍

DirectX，( Direct eXtension，简称DX) 是由微软公司创建的多媒体编程接口。由C++编程语言实现，遵循COM。被广泛使用于Microsoft Windows、Microsoft XBOX、Microsoft XBOX 360和Microsoft XBOX ONE电子游戏开发，并且只能支持这些平台。最新版本为DirectX12，创建在最新的Windows10。DirectX 是这样一组技术:它们旨在使基于Windows的让算 机成为运行和显示具有丰富多媒体元素(例如全色图形、视频、3D动画和丰富音频)的应用程序的理想平台。DirectX包括安全和性能更新程序，以及许多涵盖所有技术的新功能。应用程庄可以通过使用DirectXAPI来访问这些新功能。

DirectX加强3D图形和声音效果，并提供设计人员一个共同的硬件驱动标准，让游戏开发者不.必为每--品牌的硬件来写不同的驱动程序，也降低了用户安装及设置硬件的复杂度。从字面意义上说，Direct就是 直接的意思，而后边的x则代表了很多的意思，从这一点上可以看出DirectX的出现就是为了为众多软件提供直接服务的。

举例来说，以前在DOS下玩家玩游戏时，并不是安装上就可以玩了，他们往往首先要设置声卡的品牌和型号，然后还要设置IRQ (中断)、/Q (输入与输出)、DMA (存取模式)，如果哪.项设置的不对，那么游戏声音就发不出来。这部分的设置不仅让玩家伤透脑筋，对游戏开发者来.说就更为头痛。为了让游戏能够正确运行，开发者必须在游戏制作之初，把市面上所有声卡硬件数据都收集过来，然后根据不同的API (应用编程接口)来写不同的驱动程序。这对于游戏制作公司来说，是很难完成的，所以在当时多媒体游戏很少。微软正是看到了这个问题，为众厂家推出了一个共同的应用程序接口--DirectX。只要游戏是依照DirectX来开发的，不管显卡、声卡型号如何，统统都能玩，而且还能发挥最佳的效果。当然，前提是使用的显卡、声卡的驱动程序必须支持DirectX才行。

### CG（NVIDIA)）介绍

GLSL与HLSL分别基于OpenGL和Direct3D的接口，两者不能混用，事实上OpenGL和Direct3D一直都是冤家对头，争斗良久。OpenGL 在其长期发展中积累下的用户群庞大，这些用户会选择GLSL学习。GLSL继承了OpenGL的良好移植性，-度在unix等操作系统上独领风骚。但GLSL 的语法体系自成- -家。微软的HLSL移植性较差，在windows 平台上可谓一家独大， 这-点在很大程度上限制了HLSL的推广和发展。但是HLSL 用于DX游戏领域却是深入人心。

Cg语言(C for Graphic)是为GPU编程设计的高级着色器语言，Cg极力保留C语言的大部分语义，并让开发者从硬件细节中解脱出来，Cg同时也有-.个高级语言的其他好处，如代码的易重用性，可读性得到提高，编译器代码优化。Cg 是一个可以被OpenGL和Direct3D广泛支持的图形处理器编程语言。Cg 语言和OpenGL、DirectX 并不是同一- 层次的语言，而是OpenGL和DirectX的上层，即，Cg程序是运行在OpenGL和DirectX标准顶点和像素着色的基础上的。Cg由NVIDIA公司和微软公司相互协作在标准硬件光照语言的语法和语义上达成了-致开发。所以，HLSL和Cg其实是同一种语言。

## 图形处理器（GPU）简史

* GPU发展简史
* 可编程GPU
* GPU的优点和缺点

### GPU发展简史

GPU英文全称Graphic Processing Unit，中文翻译为“图形处理器”，在现代计算机系统中的作用变得越来越重要。

20世纪六、七十年代，受硬件条件的限制，图形显示器只是计算机输出的一种工具。限于硬件发展水平，人们只是纯粹从软件实现的角度来考虑图形用户界面的规范问题。此时还没有GPU的概念。

GPU概念在20世纪70年代末和80年代初被提出，使用单片集成电路(monolithic) 作为图形芯片，此时的GPU被用于视频 游戏和动画方面，它能够很快地进行几张图片的合成(仅限于此)。在20世纪80年代末到90年代初这段时间内，基于数字信 号处理芯片( digital signal processor chip)的GPU被研发出来，与前代相比速度更快、功能更强，当然价格是非常的昂贵。在1991年，S3Graphics公司研制出第--个单芯片2D加速器，到了1995年，主流的PC图形芯片厂商都在自己的芯片上增加了对2D加速器的支持。

1998年NVIDIA公司宣布modern GPU的研发成功，标志着GPU研发的历史性突破成为现实。通常将20世纪70年代末到. 1998年的这一段时间称之为pre-GPU时期，而自1998年往后的GPU称之为modern GPU。在pre-GPU 时期，一-些图形厂商，如SGI、Evans & Sutherland，都研发了各自的GPU，这些GPU在现在并没有被淘汰，依然在持续改进和被广泛的使用，当然价格也是非常的高昂。modern GPU使用晶体管(transistors) 进行计算，在微芯片(microchip)中，GPU所使用的晶体管已经远远超过CPU。例如，Intel 在2.4GHz的Pentium IV\_ 上使用5千5百万(55million)个晶体管:而NVIDIA 在GeForce FX GPU.上使用超过1亿2千5百万( 125 million)个晶体管，在NVIDDIA 7800 GXT.上的晶14体管达到3亿2百万(302 million)个。

回顾GPU的发展历史，自1998年后可以分为4个阶段。NVIDIA于1998年宣布Modern GPU研发成功，这标志着第一代 Modern GPU的诞生，第一代GPU包括NVIDIA TNT2，ATI 的Rage和3Dfx的Voodoo。这些GPU可以独立于CPU进行像素缓存区的更新，并可以光栅化三角面片以及进行纹理操作，但是缺乏三维顶点的空间坐标变换能力，这意味着“必须依赖于CPU执行顶点坐标变换的计算”。这-一时期的GPU功能非常有限，只能用于纹理组合的数学计算或者像素颜色值的计算。

1999年，nvidia推出 了一.款可以用“惊变"来形容的显示核心代号NV10的geforce 256。nVidia率 先将硬体T&L整合到显示核中。.T&L（T&L=Transformation[顶点变换] and Lightnig[光照处理]）原先由CPU负责,或者由另一个独立处理机处理。T&L是一 - 大进步，原因是显视核心从CPU接管了大量工作。硬件T&L引擎带来的效果是，3D模型可以用更多的多边形来描绘，这样就拥有了更加细腻的效果。而对于Lighting来说，CPU不必再计算大量的光照数据，直接通过显卡就能获得更好的效能。同时，这一阶段的GPU对于纹理的操作也扩展到了立方体纹理(cube map)。NVIDIA 的GeForce MAX，ATI 的Radeon 7500等都是在这-阶段研发的。

2001年是第三代modern GPU的发展时期，这一时期研发的GPU提供vertex programmability (顶点编程能力)，如GeForce 3,GeForce 4Ti, ATI 的8500等。这些GPU允许应用程序指定一个序列的指令进行顶点操作控制(GPU编程的本质!)，这同样是一个具有开创意义的时期，这一时期确立的GPU编程思想一直 延续到今天，不但深入到工程领域帮助改善人类日常生活(医疗、地质勘探、游戏、电影等)，而且开创或延伸了计算机科学的诸多研究领域(体绘制、光照模拟、人群动画、通用计算等)。同时，Direct8和OpenGL都本着与时俱进的精神，提供了支持vertex programmability(顶点编程能力)的扩展。不过，这一时期的GPU还不支持像素级的编程能力，即fragment programmability (片段编程能力)。

所谓Vertex，就是我们熟悉的组成3D图形的顶点，由于设计3D模型是基于坐标空间内部设计的，所以Vertex信 息包含了3D模型在空间内的坐标等信息。Vertex Shader则是对于Vertex信息的运算编程器，可以通过赋予特定的算法而在工作中改变3D模型的外形，Vertex Shader顶点运算单元可以直接检索显存中的材质数据。现在的游戏场景越来越复杂了。所涉及到的材质和多边形数量都非常惊人。顶点材质技术可以极大的提高GPU在处理复杂的游戏场景时的效率。并且游戏开发人员还可以利用Vertex Shader的这一新 的特性，充分发挥想象，实现很多非常漂亮的特效。例如在星际争霸2demo片中展示的神族母舰黑洞的技能效果。

第四代GPU的发展时期从2002年末到2003年。NVIDIA的GeForceFX和ATI Radeon 9700同时在市场的舞台，上闪亮登场，这两种GPU都支持vertex programmability(顶点编程能力)和fragment Programmability（片段编程能力）。同时DirectX和OpenGL也扩展了自身的API，用以支持vertex programmability和fragment programmability。自2003年起，可编程图形处理器正式诞生，并且由于DirectX和OpenGL锲而不舍的追赶潮流，导致基于图形硬件的编程技术，简称GPU编程，也宣告诞生。

### GPU的优越性

由于GUP具有高并行结构，所以GPU在处理图形数据和复杂算法方面拥有比CPU更高的效率。CPU大部分面积为控制器和寄存器，与之相比，GPU拥有更多的ALU（ARithmetic Logic Unit，逻辑运算单元）用于数据处理，这样的结构适合对密集型数据进行并行处理。

GPU采用流式并行计算模式，可对每个数据进行独立的并行计算，所谓“对数据进行独立计算”，即，流内任意元素的计 算不依赖于其它同类型数据，例如，计算一个顶点的世界位置坐标，不依赖于其他顶点的位置，所谓“并行计算”是指“多个数据可以同时被使用，多个数据并行运算的时间和1个数据单独执行的时间是一样的”。所以，在顶点处理程序中，可以同时处理N个顶点数据。

### GPU的缺陷

由于“任意一个元素的计算不依赖于其它同类型数据”，导致“需要知道数据之间相关性的”算法，在GPU上难以得到实现， 一个典型的例子是射线与物体的求交运算。GPU中的控制器少于CPU，致使控制能力有限。另外，进行GPU编程必须掌握计算机图形学相关知识，以及图形处理API，入门门槛较高，学习周期较长，尤其国内关于GPU编程的资料较为匮乏，这些都导致了学习的难度。在早期，GPU编程只能使用汇编语言，开发难度高、效率低，不过，随着高级Shader language的兴起，在GPU. 上编程已经容易多了。

### GPU的更多应用

科学可视化计算:由于人体CT、地质勘探、气象数据、流体力学.等科学可视化计算处理的数据量极大，仅仅基于CPU进行计算完全不能满足实时性要求，而在GPU上进行计算则可以在效率上达到质的突破，许多在CPU上非常耗时的算法，如体绘制中的光线投射算法，都可以成功移植到GPU上，所以基于GPU的科学可视化研究目前已经成为主流。 通用算法:基于GPU进行通用计算的研究逐渐成为热点，被称之为GPGPU ( General-Purpose Computing on Graphics ProcessingUnits,也被称为GPGP，或GP2)，很多数值计算等通用算法都已经在GPU\_上得到了实现，并有不俗的性能表现，目前，线性代数，物理仿真和光线跟踪算法都已经成功的移植到GPU上。在国内，中国科学院计算技术研究所进行了基于GPU的串匹配算法的实现

### 小结：

* 2003年开始正式进入可编程GPU阶段
* GPU的并行处理能力强于CPU
* 目前GPU无法替代CPU