实用性能分析

Koji Ashida
NVIDIA 公司
技术开发组

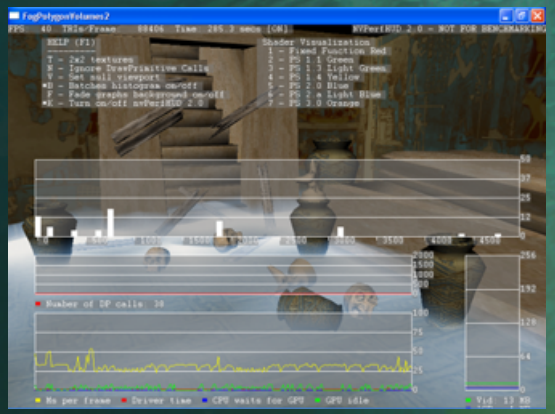
概要

- 分析工具
- 找出管线瓶颈
- 发现问题的方法

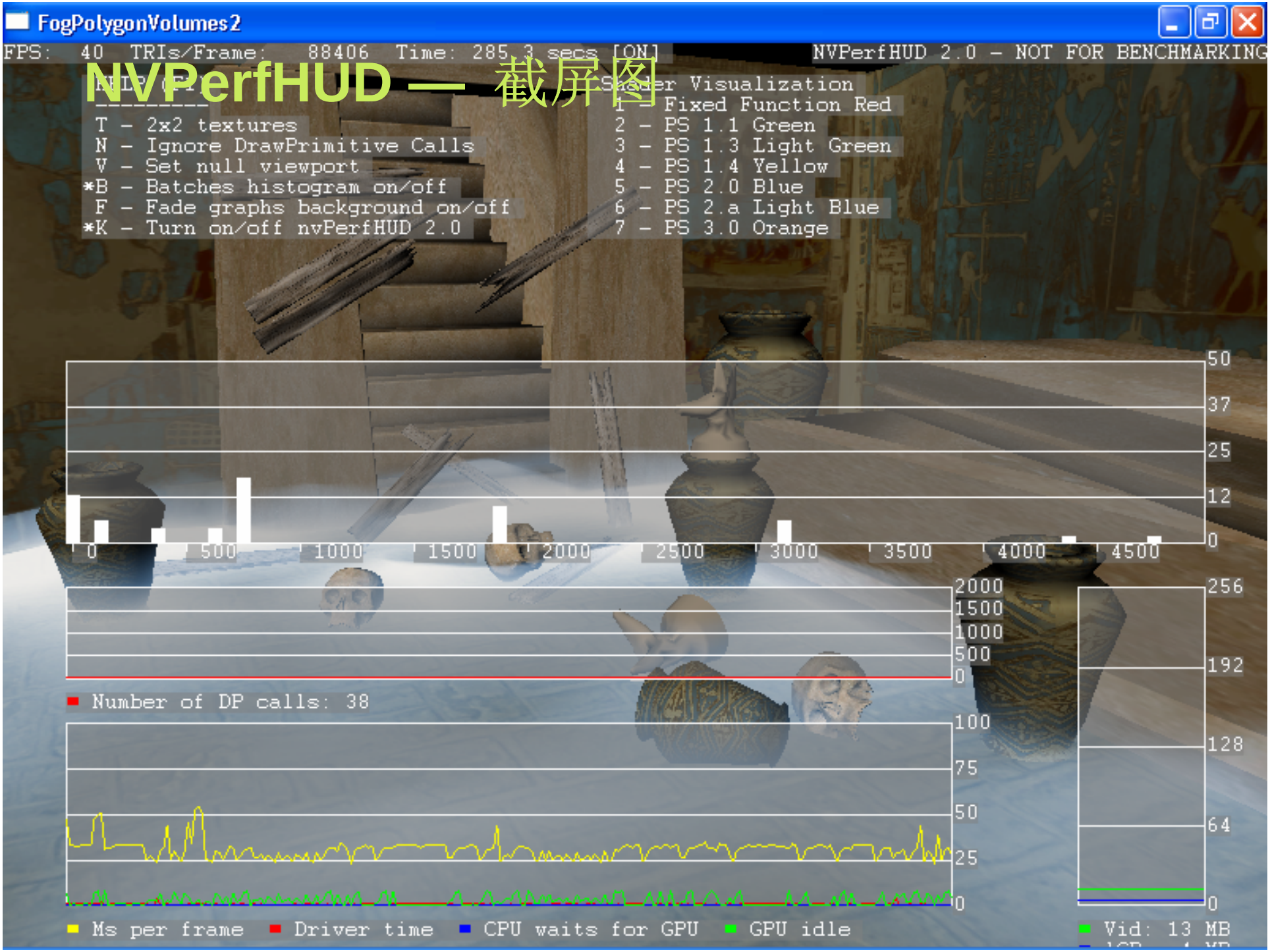
分析工具

NVPerfHUD

- 各种重要统计数据的图表覆盖
- 报告的数据包括：
  - 图形芯片空闲时间
  - 驱动程序等待时间
  - 驱动程序内时间
  - 帧时间
  - AGP / 视频内存使用率
  - # DIP/DP 调用每帧和批量柱状图
- 现在，在驱动程序之外，支持任何Direct3D9应用程序
- 以前只有注册开发人员才可以使用

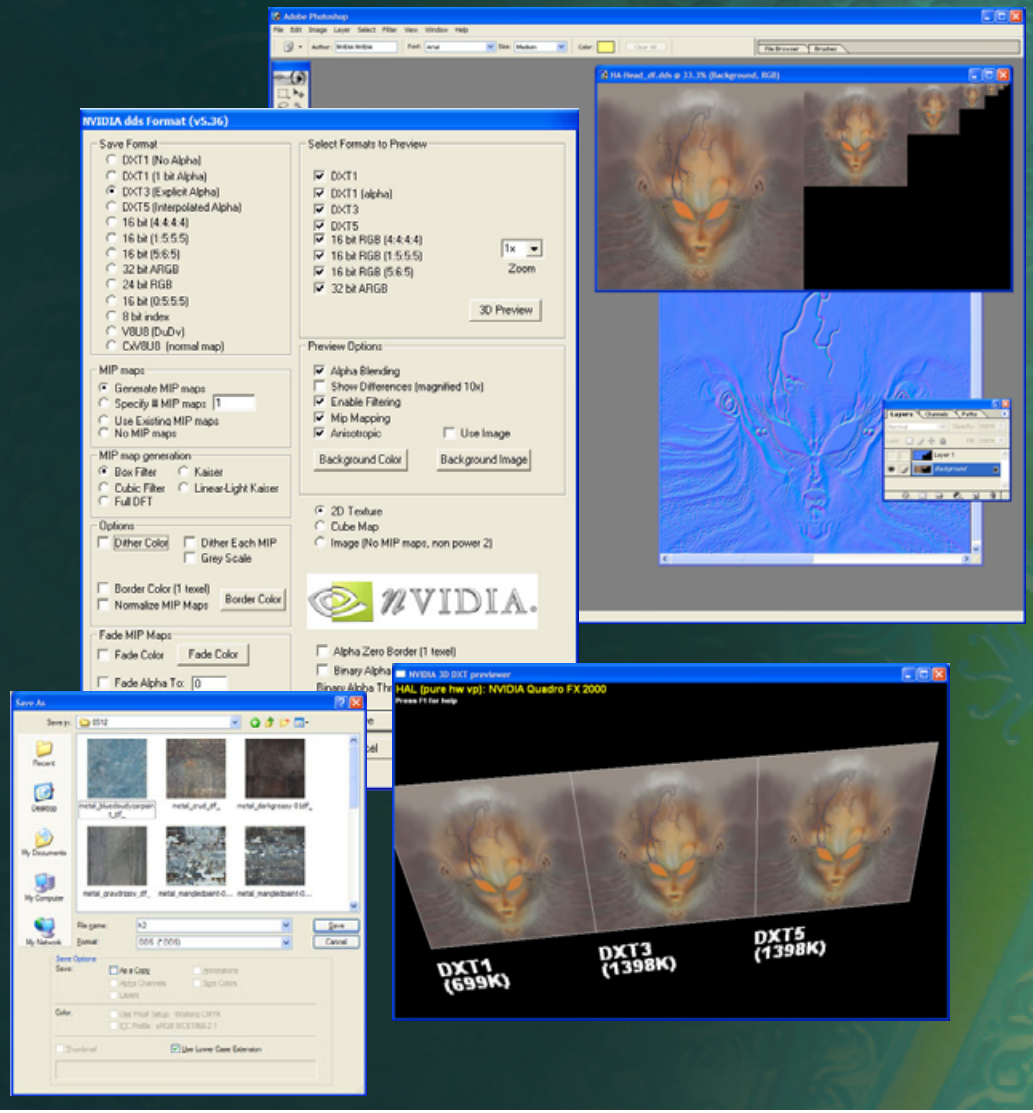


NVPerfHUD — 截屏图



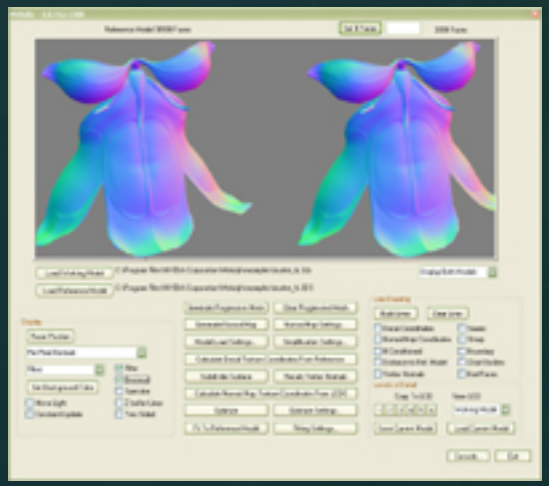
纹理工具与插件

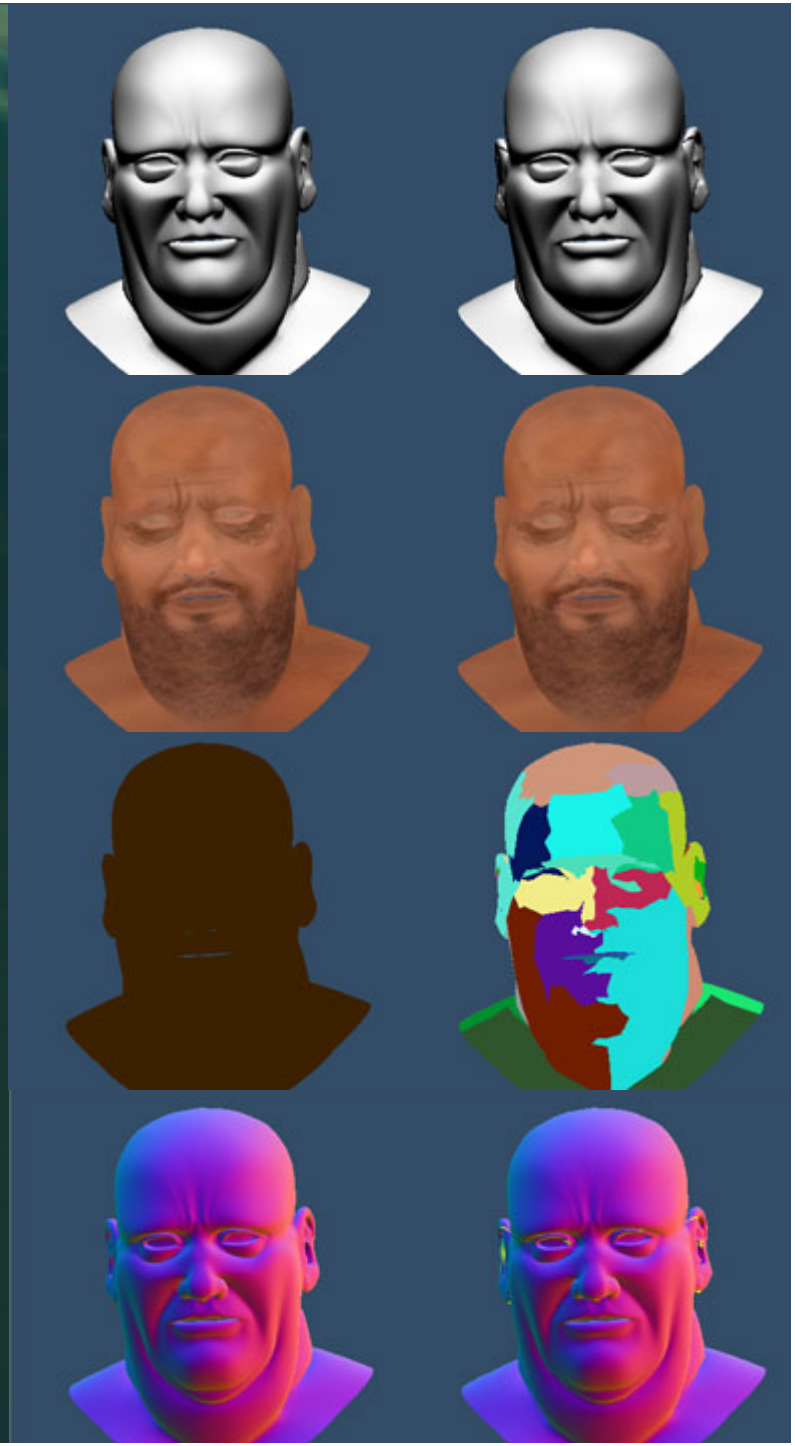
- Photoshop 插件:
  - DXT 压缩 (.dds)
  - 法线贴图创建
  - 3D 预览与差异
  - MIP 贴图生成
- 命令行与 .lib
- DDS 缩略图查看程序
- 纹理图集查看程序与创建工具



音调

- 在高解析度网格上操作 (~1.6万多边形)
- 使用高级的“十去一”渐进网格处理方式，生成LOD
- 网格优化与简化
- 基于图表的UV参数化
- Raycast法线贴图生成






Reference Model 38908 Faces
Set # Faces
2000 Faces
音调 — 截屏图
Load Working Model
C:\Program Files\NVIDIA Corporation\Melody\examples\bustier_lo.3ds
Display Both Models
Load Reference Model
C:\Program Files\NVIDIA Corporation\Melody\examples\bustier_hi.3DS
Generate Progressive Mesh
Clear Progressive Mesh
Generate Normal Map
Normal Map Settings...
Model Load Settings...
Simplification Settings...
Calculate Decal Texture Coordinates From Reference
Subdivide Surface
Recalc Vertex Normals
Calculate Normal Map Texture Coordinates From LOD0
Optimize
Optimize Settings...
Fit To Reference Model
Fitting Settings...
Display
Reset Position
Per Pixel Normals
Filled
Filter
Gouraud
Specular
Set Background Color
Z buffer Lines
Move Light
Two Sided
Constant Update
Line Drawing
Build Lines
Clear Lines
Decal Coordinates
Seams
Normal Map Coordinates
Sharp
Ill Conditioned
Boundary
Distance to Ref. Model
Chart Borders
Vertex Normals
Bad Faces
Levels of Detail
Copy To LOD
View LOD
1 2 3 4 5 6
Working Model
Save Current Model
Load Current Model

打开EXR库

- 支持.EXR图像格式
- ILM开发的HDR图像格式
  - www.openexr.org
- 每个组件的16位浮点
- .NET 2003库

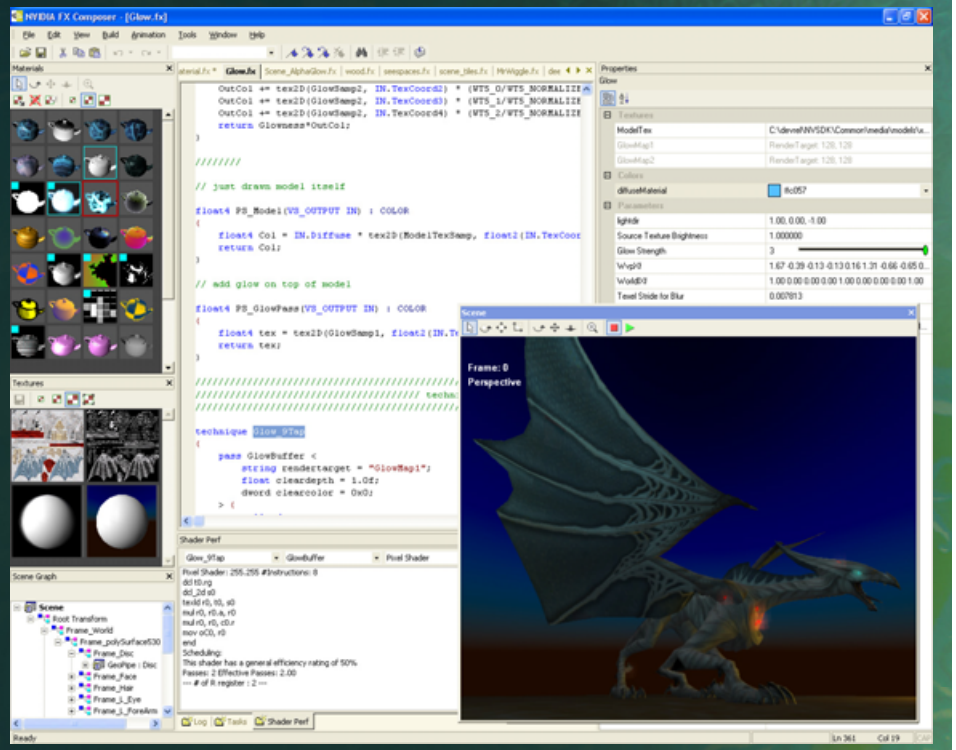
工具、库和更多……

- NVShaderPerf
  - 与FX编辑器着色器性能（Shader Perf ）面板相同的技术
  - 支持使用HLSL、!!FP1.0、!!ARBfp1.0、PS1.x和PS2.x编写的DirectX和OpenGL着色器
  - ShaderPerf为NVIDIA图形芯片的整个产品系列提供报告
- NVMeshMender
  - 修复出问题的几何图形
  - 为每个像素的光照效果准备网格
- NVTriStrip
  - “缓存激活”方式的三角剥离
  - 输出条或列表

NVIDIA FX 编辑器

FX 编辑器使开发人员可以在集成开发环境中创建高性能着色器，并能实时预览和优化。只有 **NVIDIA** 产品才能具备这些高级功能。

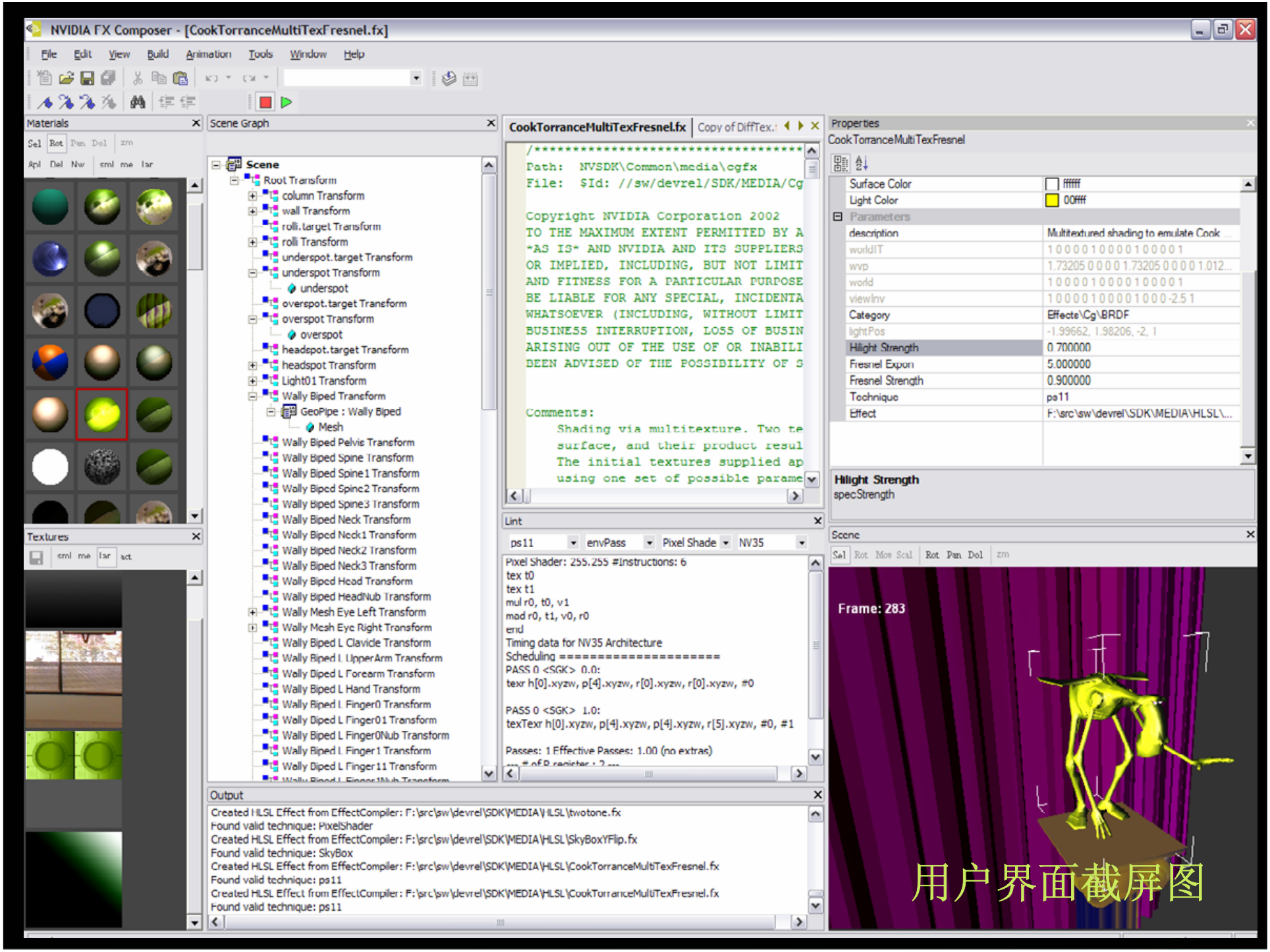
- 在功能强大的编程开发环境中创建着色器
- 利用基本的着色器纠错功能来对着色器进行纠错
- 利用高级分析和优化功能，优化着色器性能



EverQuest® content courtesy Sony Online Entertainment Inc.

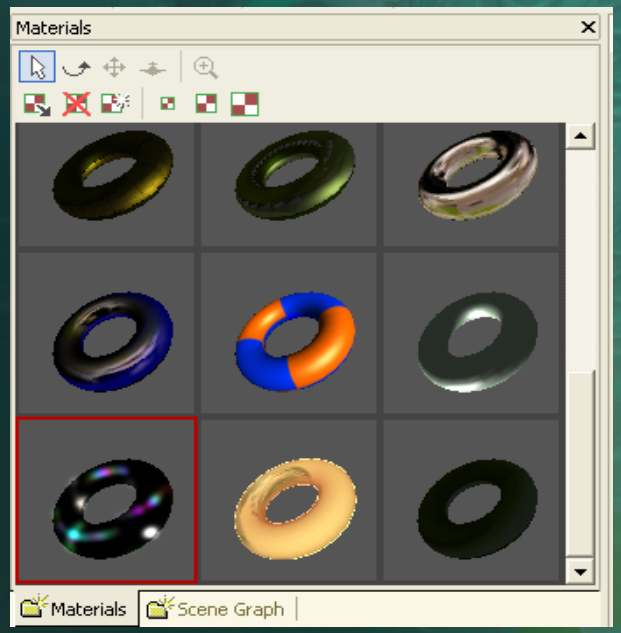

NVIDIA FX Composer - [Glow.fx]
File Edit View Build Animation Tools Window Help
Materials
Textures
Scene Graph
Scene
Root Transform
Frame_World
Frame_polySurface530
Frame_Disc
GeoPipe : Disc
Frame_Face
Frame_Hair
Frame_L_Eye
Frame_L_ForeArm
Glow.fx
Scene_AlphaGlow.fx
wood.fx
seespaces.fx
scene_tiles.fx
MrWiggle.fx
OutCol += tex2D(GlowSamp2, IN.TexCoord2) * (WT5_0/WT5_NORMALIZE
OutCol += tex2D(GlowSamp2, IN.TexCoord3) * (WT5_1/WT5_NORMALIZE
OutCol += tex2D(GlowSamp2, IN.TexCoord4) * (WT5_2/WT5_NORMALIZE
return Glowness*OutCol;
}
////////
// just drawn model itself
float4 PS_Model(VS_OUTPUT IN) : COLOR
{
float4 Col = IN.Diffuse * tex2D(ModelTexSamp, float2(IN.TexCoor
return Col;
}
// add glow on top of model
float4 PS_GlowPass(VS_OUTPUT IN) : COLOR
{
float4 tex = tex2D(GlowSamp1, float2(IN.Te
return tex;
}
technique Glow_9Tap
{
pass GlowBuffer <
string rendertarget = "GlowMap1";
float cleardepth = 1.0f;
dword clearcolor = 0x0;
Shader Perf
Glow_9Tap
GlowBuffer
Pixel Shader
Pixel Shader: 255.255 #Instructions: 8
dcl t0.rg
dcl_2d s0
texld r0, t0, s0
mul r0, r0.a, r0
mul r0, r0, c0.r
mov oC0, r0
end
Scheduling:
This shader has a general efficiency rating of 50%
Passes: 2 Effective Passes: 2.00
--- # of R register : 2 ---
Log
Tasks
Shader Perf
Properties
Glow
Textures
ModelTex
C:\devrel\NVSDK\Common\media\models\x...
GlowMap1
RenderTarget: 128, 128
GlowMap2
RenderTarget: 128, 128
Colors
diffuseMaterial
ffc057
Parameters
lightdir
1.00, 0.00, -1.00
Source Texture Brightness
1.000000
Glow Strength
3
WvpXf
1.67 -0.39 -0.13 -0.13 0.16 1.31 -0.66 -0.65 0...
WorldXf
1.00 0.00 0.00 0.00 1.00 0.00 0.00 0.00 1.00
Texel Stride for Blur
0.007813
Scene
Frame: 0
Perspective
Ready
Ln 361
Col 19

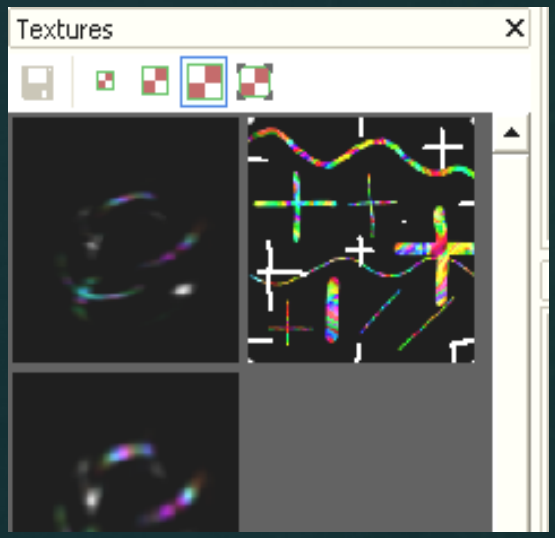
FX编辑器 — 截屏图


NVIDIA FX Composer - [CookTorranceMultiTexFresnel.fx]
File Edit View Build Animation Tools Window Help
Materials
Textures
Scene Graph
Scene
Root Transform
column Transform
wall Transform
rolli.target Transform
rolli Transform
underspot.target Transform
underspot Transform
underspot
overspot.target Transform
overspot Transform
overspot
headspot.target Transform
headspot Transform
Light01 Transform
Wally Biped Transform
GeoPipe : Wally Biped
Mesh
Wally Biped Pelvis Transform
Wally Biped Spine Transform
Wally Biped Spine1 Transform
Wally Biped Spine2 Transform
Wally Biped Spine3 Transform
Wally Biped Neck Transform
Wally Biped Neck1 Transform
Wally Biped Neck2 Transform
Wally Biped Neck3 Transform
Wally Biped Head Transform
Wally Biped HeadNub Transform
Wally Mesh Eye Left Transform
Wally Mesh Eye Right Transform
Wally Biped L Clavicle Transform
Wally Biped L UpperArm Transform
Wally Biped L Forearm Transform
Wally Biped L Hand Transform
Wally Biped L Finger0 Transform
Wally Biped L Finger01 Transform
Wally Biped L Finger0Nub Transform
Wally Biped L Finger1 Transform
Wally Biped L Finger11 Transform
CookTorranceMultiTexFresnel.fx
Copy of DiffTex.
/**********
Path: NVSDK\Common\media\cgfx
File: $Id: //sw/devrel/SDK/MEDIA/Cg
Copyright NVIDIA Corporation 2002
TO THE MAXIMUM EXTENT PERMITTED BY A
"AS IS" AND NVIDIA AND ITS SUPPLIERS
OR IMPLIED, INCLUDING, BUT NOT LIMIT
AND FITNESS FOR A PARTICULAR PURPOSE
BE LIABLE FOR ANY SPECIAL, INCIDENTA
WHATSOEVER (INCLUDING, WITHOUT LIMIT
BUSINESS INTERRUPTION, LOSS OF BUSIN
ARISING OUT OF THE USE OF OR INABILI
BEEN ADVISED OF THE POSSIBILITY OF S
Comments:
Shading via multitexture. Two te
surface, and their product resul
The initial textures supplied ap
using one set of possible parame
Lint
ps11 envPass Pixel Shade NV35
Pixel Shader: 255.255 #Instructions: 6
tex t0
tex t1
mul r0, t0, v1
mad r0, t1, v0, r0
end
Timing data for NV35 Architecture
Scheduling ==========
PASS 0 <SGK> 0.0:
texr h[0].xyzw, p[4].xyzw, r[0].xyzw, r[0].xyzw, #0
PASS 0 <SGK> 1.0:
texTexr h[0].xyzw, p[4].xyzw, p[4].xyzw, r[5].xyzw, #0, #1
Passes: 1 Effective Passes: 1.00 (no extras)
Output
Created HLSL Effect from EffectCompiler: F:\src\sw\devrel\SDK\MEDIA\HLSL\twotone.fx
Found valid technique: PixelShader
Created HLSL Effect from EffectCompiler: F:\src\sw\devrel\SDK\MEDIA\HLSL\SkyBoxYFlip.fx
Found valid technique: SkyBox
Created HLSL Effect from EffectCompiler: F:\src\sw\devrel\SDK\MEDIA\HLSL\CookTorranceMultiTexFresnel.fx
Found valid technique: ps11
Created HLSL Effect from EffectCompiler: F:\src\sw\devrel\SDK\MEDIA\HLSL\CookTorranceMultiTexFresnel.fx
Found valid technique: ps11
Properties
CookTorranceMultiTexFresnel
Surface Color ffffff
Light Color 00ffff
Parameters
description Multitextured shading to emulate Cook
worldIT 1 0 0 0 0 1 0 0 0 0 1 0 0 0 0 1
wvp 1.73205 0 0 0 0 1.73205 0 0 0 0 1.012...
world 1 0 0 0 0 1 0 0 0 0 1 0 0 0 0 1
viewInv 1 0 0 0 0 1 0 0 0 0 1 0 0 0 -2.5 1
Category Effects\Cg\BRDF
lightPos -1.99662, 1.98206, -2, 1
Hilight Strength 0.700000
Fresnel Expon 5.000000
Fresnel Strength 0.900000
Technique ps11
Effect F:\src\sw\devrel\SDK\MEDIA\HLSL\...
Hilight Strength
specStrength
Scene
Frame: 283
用户界面截屏图

材料与纹理

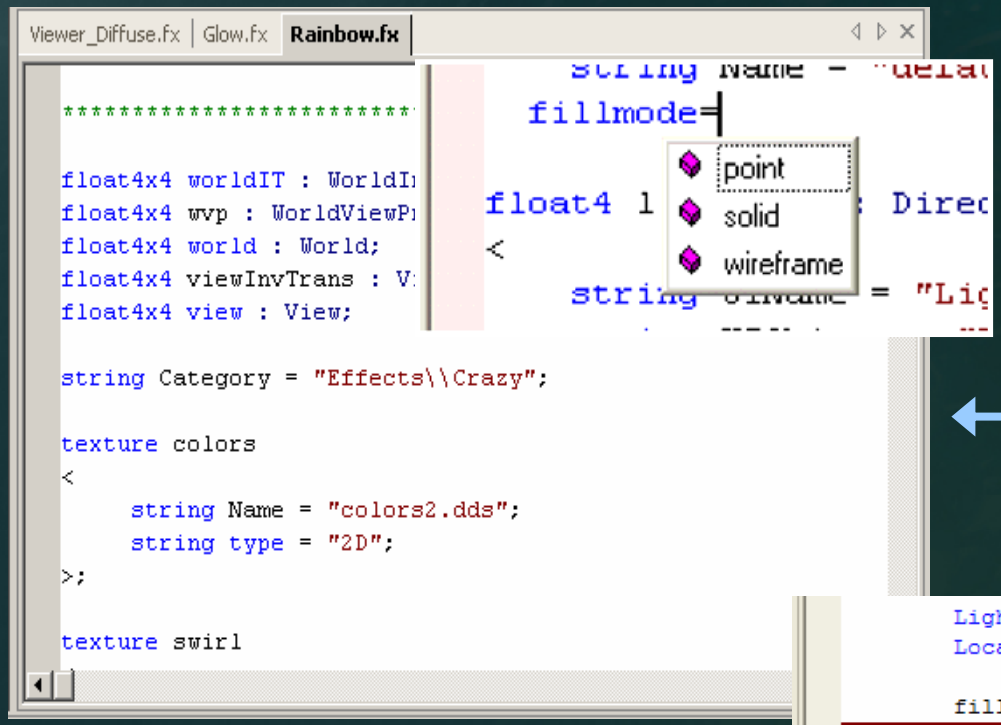
- 在您的场景内同时预览所有的FX文件
- 在场景面板内，将它们应用到场景中适当的部分



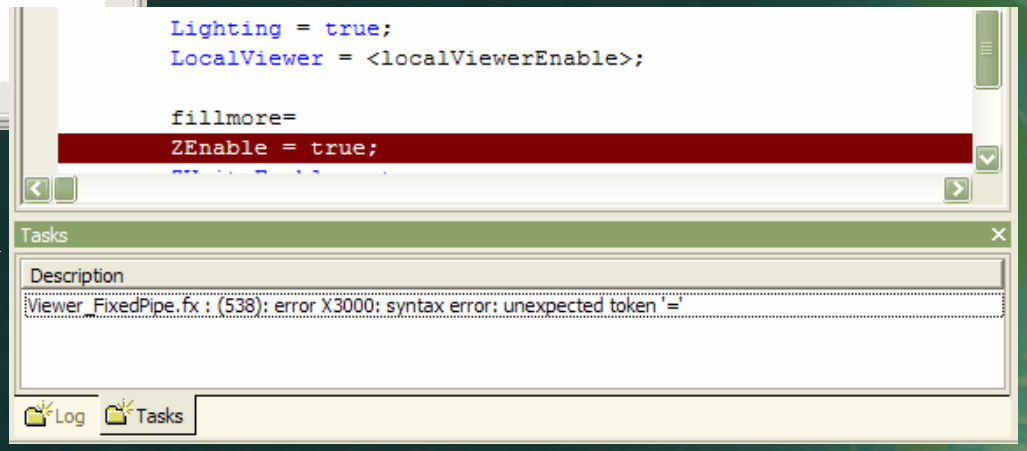


- 查看源纹理
- 预览渲染对象
- 将纹理保存到磁盘！

编辑与纠错

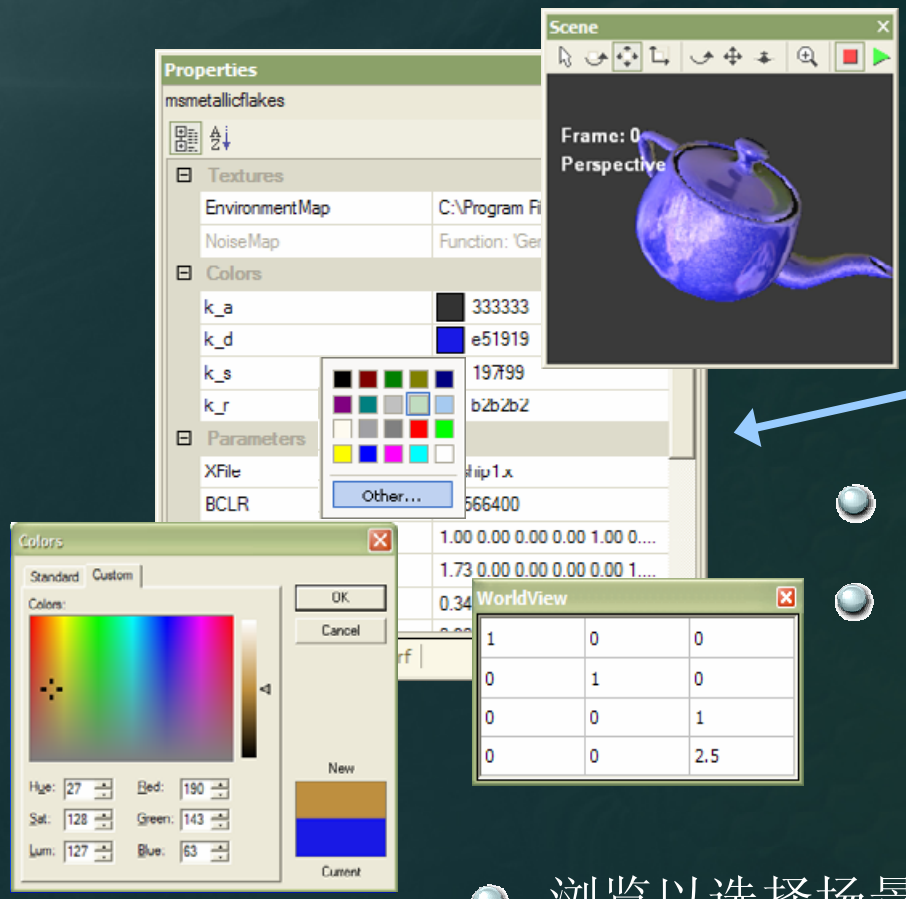


- 编辑多个.FX文件
- 智能感知（自动完成）
- 语法凸显

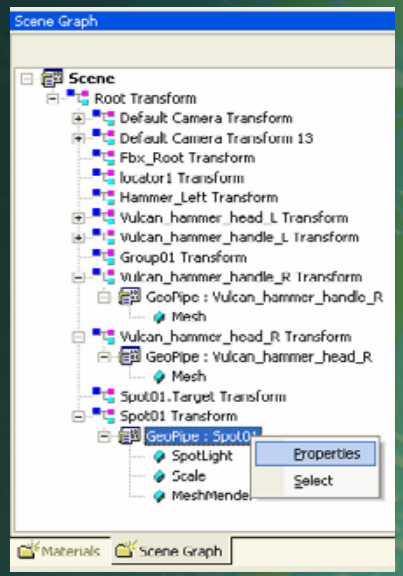


- “跳至错误” 使您能够快速地找到和修复错误

预览与自定义

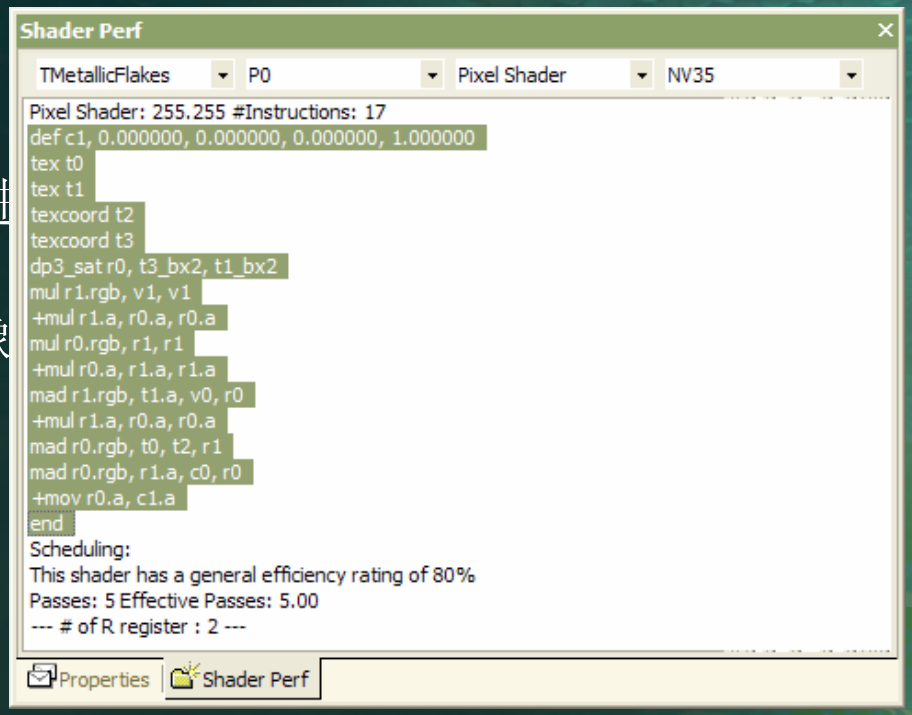


- 便于编辑着色器参数
- 自动解析语义与注释
- 快速选择自定义颜色值
- 使用可撕开的对话框（tear-off dialogs）更新矢量和矩阵值
- 浏览以选择场景元素，并在场景图表面板里编辑属性

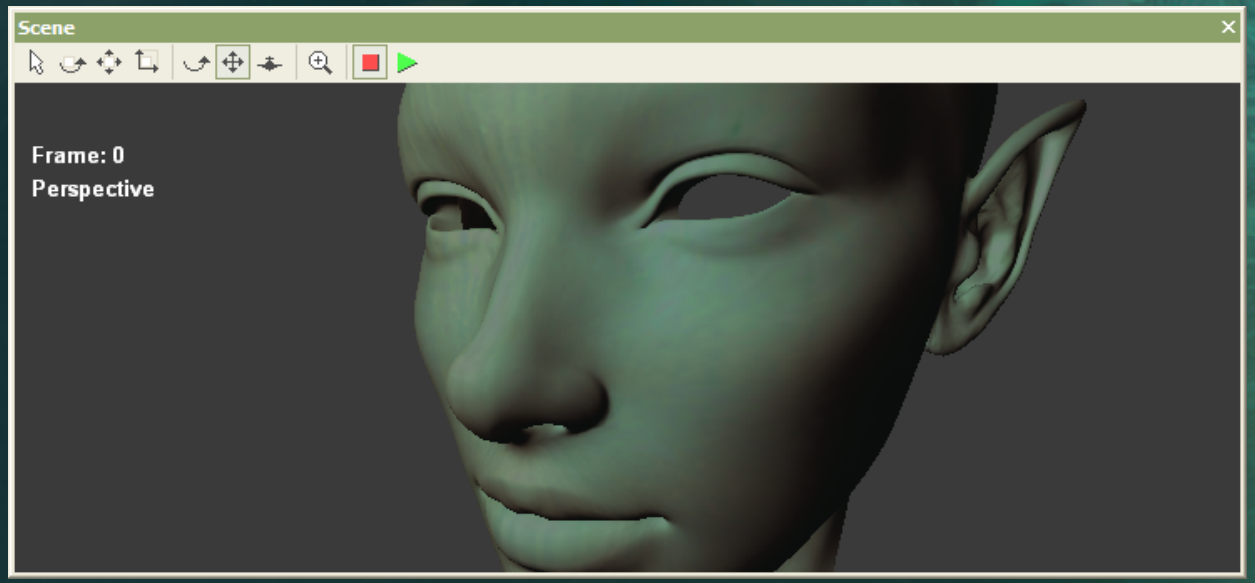


优化着色器性能

- 使用着色器性能（Shader Perf ）面板
- 选择技术、流程、以及顶点或像素进行分析
- 模拟任何新 NVIDIA 图形芯片上的像素着色器性能
- 优化的 DirectX 组件
- NVIDIA 性能表现分析
  - 图形芯片周期计算
  - 效率/利用率
  - 流程数
  - 寄存器使用率



NVIDIA FX 编辑器



场景面板

- 实时预览3D场景
  - 将材料应用到场景元素中
  - 操作场景元素或整个场景
  - 使用基本要素或导入.x模型和.nvb场景
  - 设定您自己的关键帧或播放现有的动画
  - 设定光源，自定义光照属性
  - 选择用户定义的镜头或默认场景镜头

常见问题解答

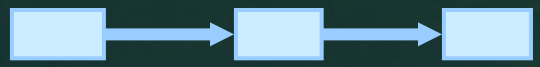
- 是否支持Cg或GLSL？
- 是否可以与我的引擎整合？
- 是否与DCC应用程序整合？
- 与RenderMonkey、特效编辑或其他工具相比，效果如何？
- 支持其他平台吗？

查询最新版本的FX编辑器和全部常见问题解答，请浏览：
http://developer.nvidia.com/fxcomposer

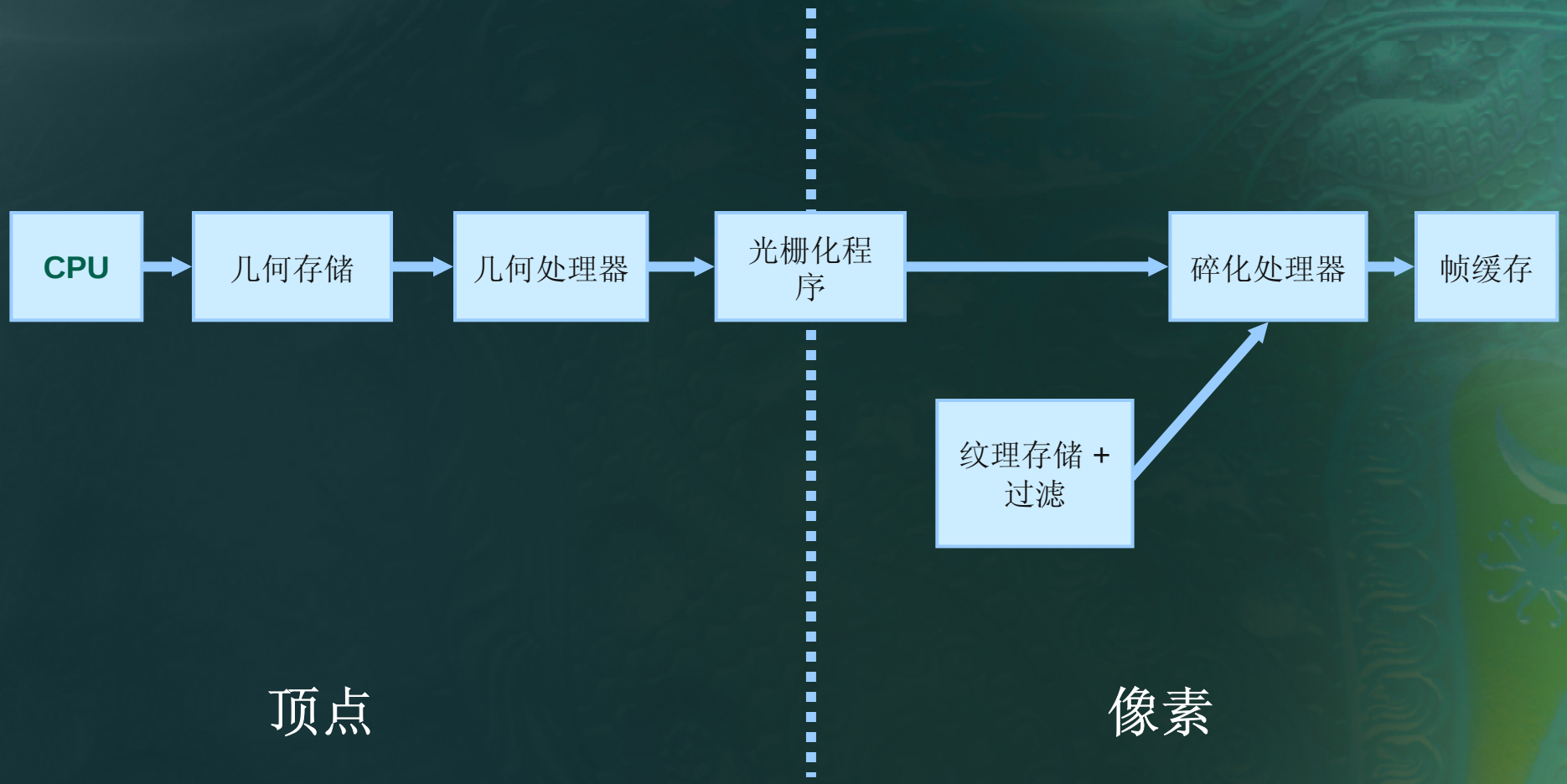
性能优化

基本原理

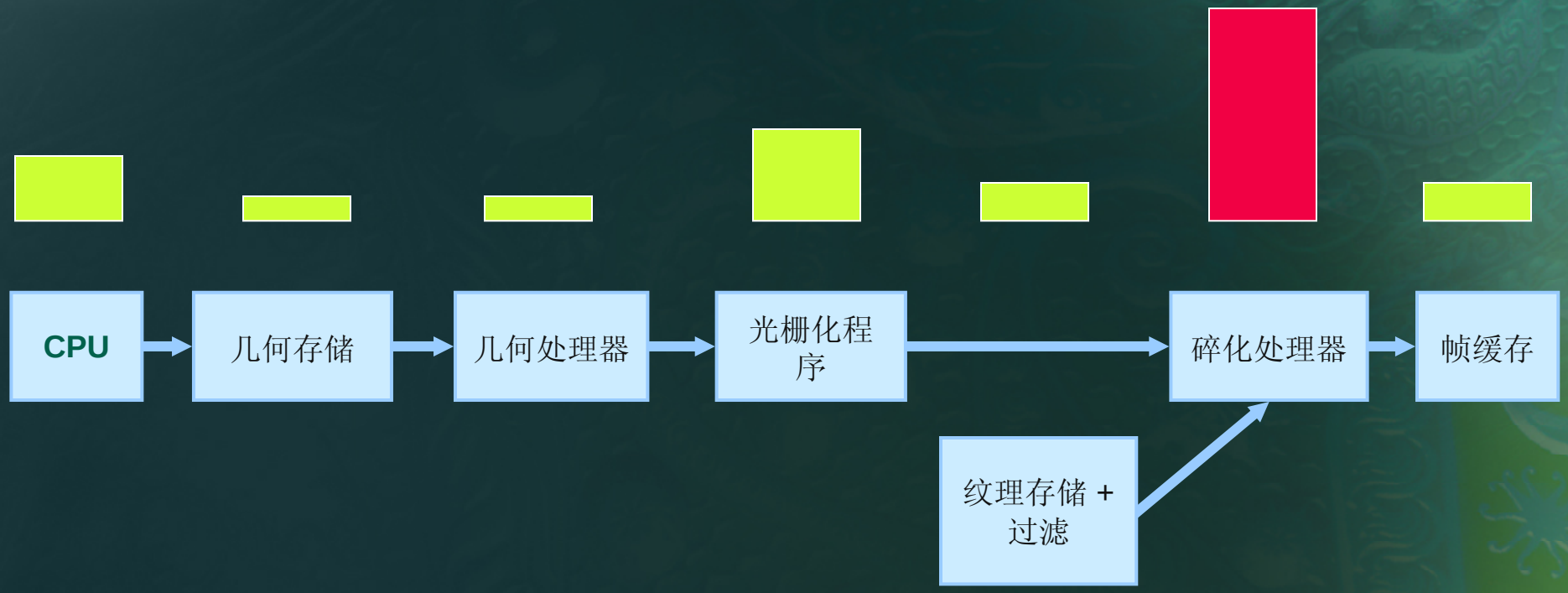
- 管线架构
- 瓶颈辨识和消除
- 平衡管线



管线架构



糟糕的瓶颈

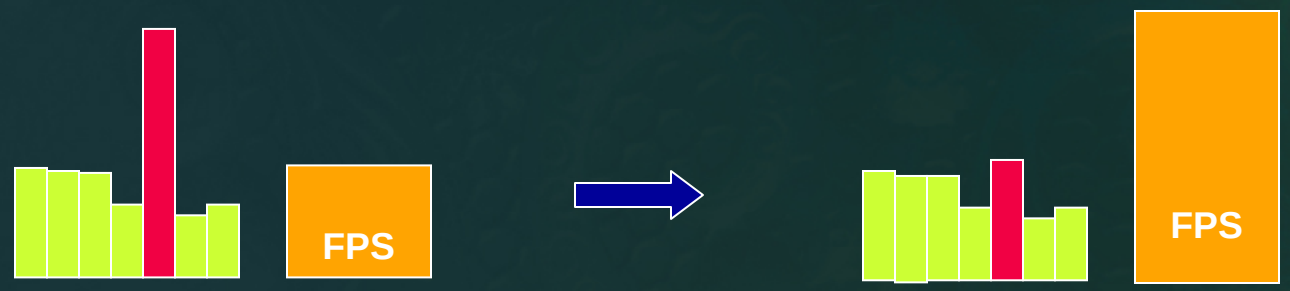


瓶颈辨识

- 有两个方法可以辨识瓶颈
- 修改该阶段
- 排除其他阶段

瓶颈辨识

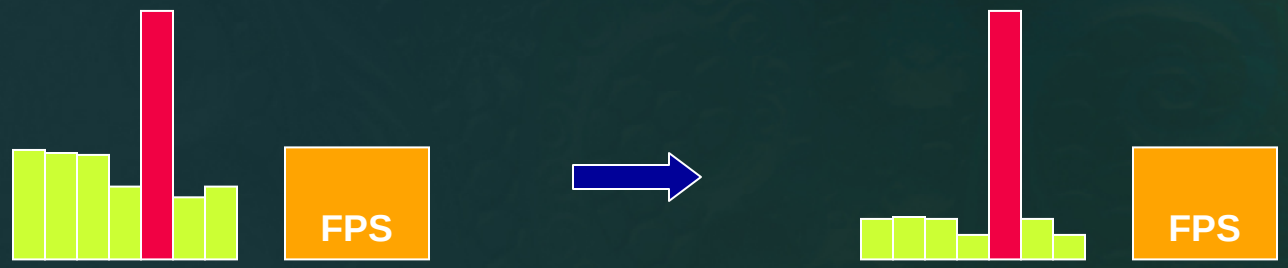
- 修改该阶段
  - 通过降低其工作负荷



  - 如果性能显著改善，那么很明显，该阶段就是瓶颈
  - 注意不要改变其他阶段的工作负荷！

瓶颈辨识

- 排除其他阶段
  - 大大降低其他阶段的工作负荷，或让它们停止工作



  - 如果性能并未显著改变，那么很明显，该阶段就是瓶颈
  - 注意不要改变这个阶段的工作负荷！

瓶颈辨识

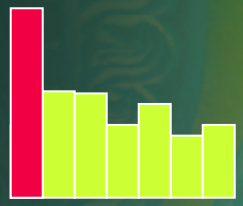
- 大多数情况下，任何阶段发生改变都会影响到其他阶段
- 很难确定究竟应该进行哪项测试
- 让我们探讨一下部分测试项目

瓶颈辨识：CPU

- 有问题?
- 有可能是游戏的问题
  - 复杂的动作、人工智能、游戏逻辑等
  - 内存管理
  - 数据结构
- 可能是因为API不正确的使用
  - 检查纠错运行时间输出，查找错误和警告
- 可能是显示器驱动程序出问题
  - 太多批处理命令

瓶颈辨识：CPU

- 降低CPU的工作负荷
- 暂时关闭
  - 游戏逻辑
  - 人工智能
  - 动作
  - 任何不改变渲染工作负荷而又占用大量CPU资源的东西



瓶颈辨识：CPU

- 排除其他阶段
- 关闭DrawPrimitive（绘制基本要素）调用指令
  - 正常设置所有东西，但是当要进行渲染时，不要使用DrawPrimitive*调用指令
  - 问题：当发出“绘制基本要素”调用指令之后，您不知道运行时间或驱动程序上发生了什么样的动作
- 使用 VTUNE 或 NVPerfHUD

瓶颈辨识：顶点

- 问题?
- 将顶点和索引转移到卡上
- 将顶点和索引转变为三角
- 顶点高速缓存丢失
- 使用占用大量资源的顶点着色器

瓶颈的辨识：顶点

- 降低顶点消耗
- 使用更简单的顶点着色器
- 发送更少的三角??
  - 不好
- 降低AGP孔径 ??
  - 可能不好
  - 使用NVPerfHUD来检查视频内存
  - 如果内存已满，可能有AGP纹理

瓶颈辨识：顶点

- 排除其他阶段
- 渲染到更小的后台缓存；这样可以排除
  - 纹理、帧缓存、像素着色器
- 测试CPU瓶颈
- 也能够渲染到更小的视窗而不是更小的后台缓存。仍然排除
  - 纹理、帧缓存、像素着色器

瓶颈辨识：光栅化程序

- 很少成为瓶颈，应该先花点时间测试其他阶段

瓶颈辨识：纹理

- 有问题?
- 纹理高速缓存丢失
- 巨大纹理
- 带宽
- 纹理超出 AGP

瓶颈辨识：纹理

- 降低纹理带宽
- 使用小型（2x2）纹理
  - 很好，但是如果您对纹理 alpha 使用 alpha 测试，那么电脑可能运行得更加缓慢，因为填充量增大了。不过这仍然是个很好的测试。
- 如果还没有准备好，那么使用 mip 贴图
- 如果各向异性过滤功能已打开，那么关闭它

瓶颈辨识：纹理

- 排除其他阶段
- 因为纹理很容易直接测试，我们建议您依靠这个手段

瓶颈辨识：片段

- 有问题?
- 占用大量资源的像素着色器
- 渲染超过需要量的片段
  - 高深度复杂性
  - 不佳的 z-cull

瓶颈辨识：片段

- 改变该阶段
- 输出一个实心颜色
  - 好：没有在每个片段上工作
  - 但也影响纹理，所以您必须排除纹理
- 使用更简单的数学运算
  - 好：每个片段做更少的工作
  - 但要确保运算时索引纹理的方式不变，否则，您将需要同时改变纹理阶段

瓶颈辨识：FB

- 有问题?
- 以超出需要的次数操作缓存
  - 多流程
- 太多alpha混合
- 利用了太大的缓存
  - 在不需要的时候应用了模版
  - 很多时候，动态反射立体贴图可以使用r5g6b5色彩，而不是x8r8g8b8色彩

瓶颈辨识：FB

- 改变该阶段
- 使用16位深度缓存，而不是24位的
- 使用16位色彩缓存，而不是32位的

实际操作

实际操作：清理机器

- 使用正确的驱动程序
- 释放游戏程序（打开优化功能）
- 检查纠错输出，查看警告或错误，但……
- 使用释放 D3D 运行时间 !!!
- 无最大有效性
- 没有使驱动程序过载的各向异性过滤或抗锯齿功能
- 关闭了 V-sync

实际操作：例1

实际操作：例1

- 动态顶点缓存
  - 不佳的创建旗标

```
HRESULT hr = pd3dDevice->CreateVertexBuffer(
                    6* sizeof( PARTICLE_VERT ),
                    0,      //declares this as static
                    PARTICLE_VERT::FVF,
                    D3DPOOL_DEFAULT,
                    &m_pVB,
                    NULL );
```

实际操作：例1

- 动态顶点缓存
  - 良好的创建旗标

```
HRESULT hr = pd3dDevice->CreateVertexBuffer(
                    6* sizeof( PARTICLE_VERT ),
                    D3DUSAGE_DYNAMIC |
                    D3DUSAGE_WRITEONLY,
                    PARTICLE_VERT::FVF,
                    D3DPOOL_DEFAULT,
                    &m_pVB,
                    NULL );
```

实际操作：例1

- 动态顶点缓存
  - 不佳的锁定旗标

```
m_pVB->Lock(0, 0,(void**)&quadTris, 0);
```

- 完全没有旗标 !?
  - 这可不怎么好……

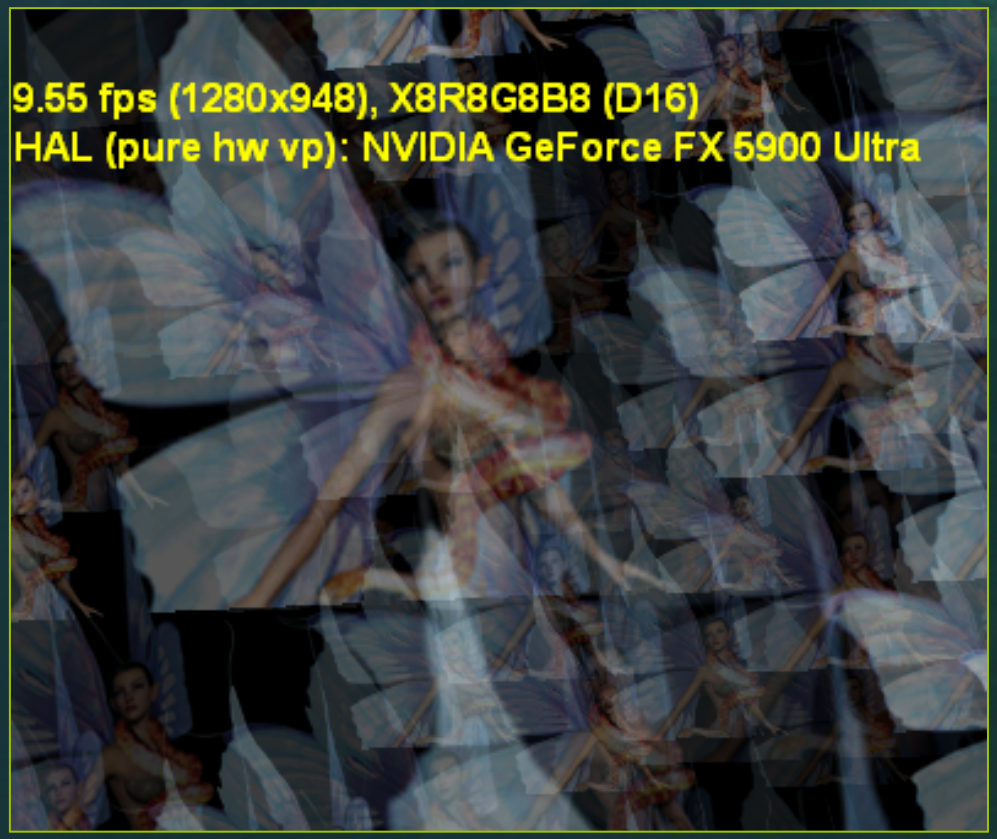
实际操作：例1

- 动态顶点缓存
  - 良好的锁定旗标

```
m_pVB->Lock(0, 0,(void**)&quadTris,
D3DLOCK_NOSYSLOCK | D3DLOCK_DISCARD);
```

- 首次锁定每帧顶点缓存时，使用 D3DLOCK_DISCARD
  - 当缓存充满之后再次使用
  - 否则，请只使用 NOSYSLOCK

实际操作：例2



实际操作：例2

- 纹理带宽占用过多
- 使用 mip 贴图
- 可能的话，请使用 dxt1
  - 有些显卡可以在高速缓存中储存压缩数据
- 在不影响效果的情况下，使用更细的纹理
  - 草叶是否真的需要 1024x1024 的纹理？-- 也许

实际操作：例3

实际操作：例3

- 占用大量资源的像素着色器
- 可以表现大型特效
- 仅3个顶点，但可能是上百万个像素
  - 只能使用 1024x1024

就看像素 !!

实际操作：例3

- 36 个循环不好

Shader Perf

TestFXCheapVSI | p0 | Pixel Shader | GeForceFX 595C

```
*****************************************
Target: GeForceFX 5950 (NV38) :: Unified Compiler: v56.58
Cycles: 36 :: # R Registers: 4
GPU Utilization: 54.00%
A large number of registers are being used which are causing register file stalls
*****************************************
PS Instructions: 45
```

实际操作：例3

- 11 个循环不好

Shader Perf

TestFXCheapVSI | p0 | Pixel Shader | GeForceFX 595C

```
*****************************************
Target: GeForceFX 5950 (NV38) :: Unified Compiler: v56.58
Cycles: 11 :: # R Registers: 2
GPU Utilization: 54.00%
*****************************************
PS Instructions: 26
```

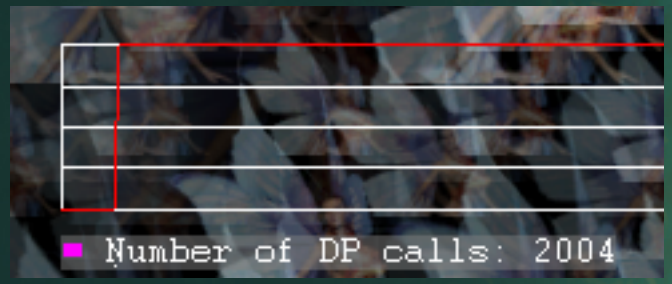
实际操作：例3

- 改变了什么?
- 将三角上的常量运算移动到顶点着色器内
- 使用 “半” 而不是 “浮”
- 在不需要的时候，不使用 “规范化”
  - 参见 “规范化的启发”
  - http://developer.nvidia.com

实际操作：例4

实际操作：例4

- 太多批处理命令
- 每四边形作为其自身批进行发送
- 相反，将所有四边形作为一个大VB，利用一个调用命令发送

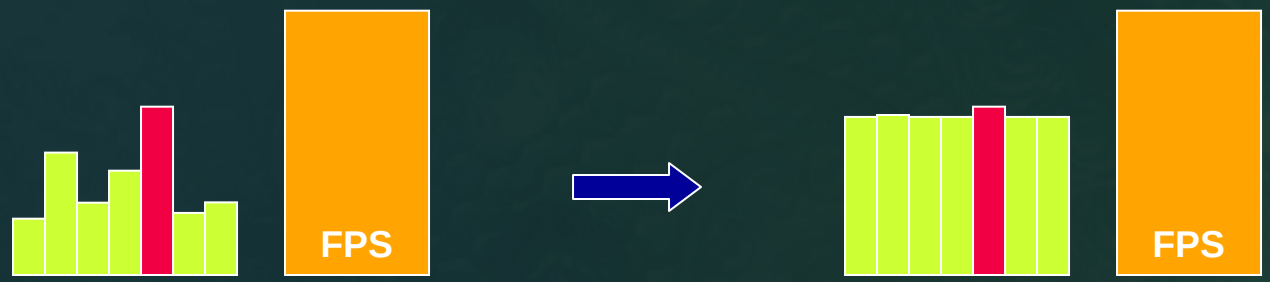


实际操作：例4

- 如果它们使用不同的纹理会怎样?
- 使用纹理图集
- 将两个纹理放在一个纹理内，使用一个顶点着色器和一个像素着色器来弥补纹理坐标

平衡管线

- 更多地使用未受瓶颈限制的阶段，以此来平衡管线
- 注意不要过多地使用这些阶段



总结

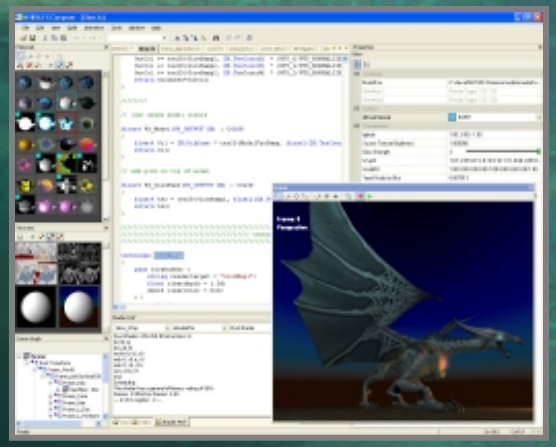
- NVIDIA 提供多种性能分析工具
- 管线架构受瓶颈制约
- 利用快速测试来辨识瓶颈
- 利用 NVPerfHUD 来分析管线
- 利用 FX 编辑器来帮助优化着色器

如需更多信息……

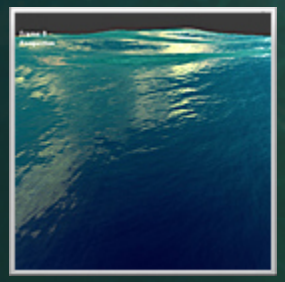
- 请下载演示文稿、软件开发工具包（SDK）和工具等，网址：http://developer.nvidia.com
- 关于工具和软件开发工具包（SDK）的问题、要求及评论等，请电邮：sdkfeedback@nvidia.com
- 了解本演示文稿的详情，请电邮: kashida@nvidia.com

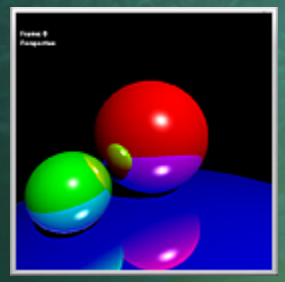
developer.nvidia.com
图形芯片编程之源

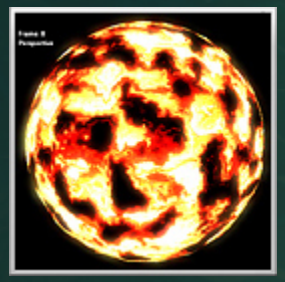
- 最新文件
- 软件开发工具包（SDK）
- 前沿工具 tools
  - 性能分析工具
  - 内容创建工具
- 数百个特效
- 视频演示文稿与指南
- 库与基本工具
- 新闻和新闻通讯档案库

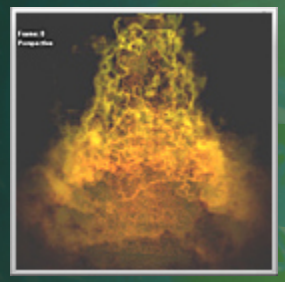


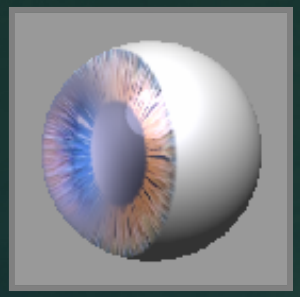
EverQuest® content courtesy Sony Online Entertainment Inc.











NVIDIA SDK

实时编程开发人员的资源库

数百个源代码样例与特效，

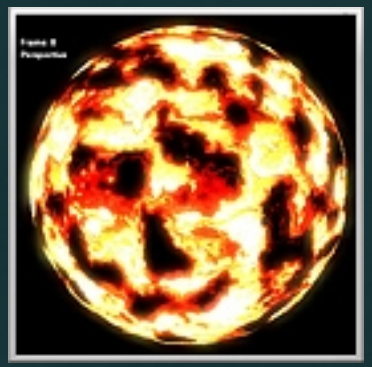
可以帮助您充分利用图形科技领域的最新成果

- 无数最新的 DirectX 和 OpenGL 代码样例，完整的源代码，大量的帮助内容：

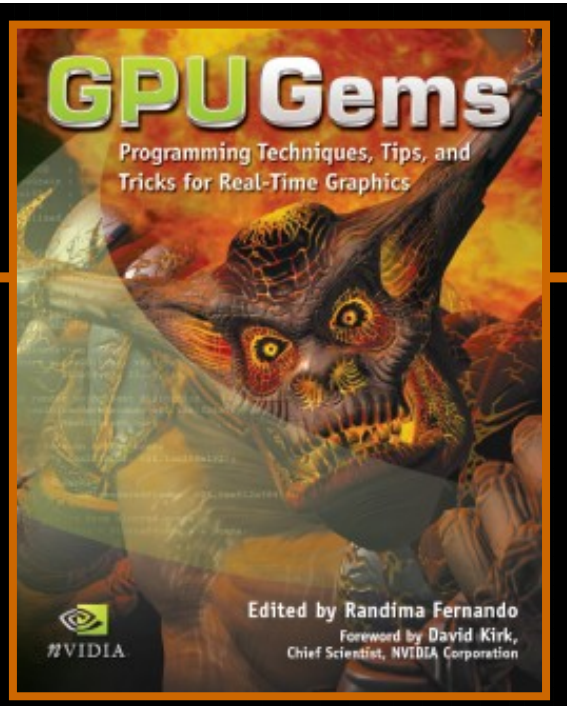
  几何实例引用、彩虹、雾虹、Blood着色器、透视阴影贴图、纹理图集工具……

- 数百个特效，带自定义几何图形、动画和更多内容：

  皮肤、塑胶、火焰/火苗、闪光、Gooch、图像过滤器、HLSL 纠错技巧、 纹理 BRDF、纹理色调映射，甚至还带有简单的光线追踪器！



《图形芯片精华》：编程技术、提示、以及处理实时图形的技巧



- 来自顶级公司和大学专家的实用的实时图形处理技巧
- 物超所值：
  - 来自业内专家的技术
  - 全彩色（300 多个图表和截屏图）
  - 硬封面
  - 816 页

查询更多详情，请浏览：
http://developer.nvidia.com/GPUGems

"《*图形芯片精华*》汇集了各种高级图形处理技巧。编程新手和图形专家均能从中受益匪浅。"

Tim Sweeney
史诗（Epic）游戏公司*虚幻*（*Unreal*）游戏的主要编程人员

"本书收集了很多文章，其内容在广度和深度上都有较大突破。书中既有面向产品的案例研究，也有先前未公开过的前沿研究成果和详实的指南等。另外，本书中还有大量的源代码样例和演示（demos）等"

Eric Haines
《*实时渲染*》的作者

Cg 工具包

- NVIDIA Cg 编译器
  - 顶点 (DirectX 9, OpenGL 1.4)
  - 像素 (DirectX 9)
- Cg 标准库
- DirectX 和 OpenGL 的 Cg 运行时间库
- NVIDIA Cg 浏览器
- Cg 语言说明
- Cg 用户手册
- Cg着色器
  （预先编写好的分类程序）
- The Cg 指南
  (developer.nvidia.com/CgTutorial)