

# 从Flash 2D走向3D世界



**郑立松**

Albert Zheng <lisong.zheng@gmail.com>

2012年12月



# 章节大纲

1

开头篇

2

技术篇

3

引擎和工具篇



# 开头篇



# Flash Stage3D目前的能力



<http://member.square-enix.com/jp/gamezone/legendworld/battleflash/index.php>

开发商：日本Square Enix(史克威尔艾尼克斯)，《最终幻想》的厂商

# Flash 3D页游目前的市场状况



- Flash 3D页游：页游领域的下一个增长点
- 市场成熟的条件：技术 + 大作 + 用户习惯
  - Flash Stage3D技术已熟，但市场还缺乏大作
  - Unity3D实现的页游也在“帮助”培育3D市场和用户习惯 🤔

| 状态    | 厂家                                       | 游戏           | 类型          | 备注      |
|-------|--|--------------|-------------|---------|
| 已上线运营 | 广州九艺                                     | 封神无双         | 全3D；MMOARPG |         |
| 内测中   | 上海一家公司                                   | 深渊           | 3D、锁定视角     |         |
|       | 上海松果工场                                   | 格斗高手         | 全3D；格斗类型    |         |
|       | 杭州无端科技                                   | 生死狙击         | 3D FPS      |         |
| 体验版   | 日本Square Enix<br>(史克威尔艾尼克斯)<br>《最终幻想》的厂家 | Legend World | 全3D；MMOARPG | 已达游戏机水准 |
| 开发中   | 呈天游                                      | ?            | 3D FPS      |         |

# Stage3D技术的典型应用场景



- 掌握Flash技术最高点：拨高门槛、想降低，游刃有余
- 根据市场需要，灵活将Stage3D技术应用到各类产品中

| 产品类型   | 视角     | 场景缩放 | 精灵渲染方式                     | 场景渲染方式                   | 游戏实例 | 厂商    |
|--------|--------|------|----------------------------|--------------------------|------|-------|
| 全3D游戏  | 全视角无锁定 | 支持   | 3D模型 + 3D动画                | 3D模型                     | 封神无双 | 广州九艺  |
| 3D游戏   | 视角锁定   | 不支持  | 3D模型 + 3D动画                | 3D模型                     | 深渊   |       |
| 2.5D游戏 | 视角锁定   | 不支持  | 3D模型 + 3D动画                | 2D位图                     | 天界   | 趣游    |
|        |        |      |                            |                          | ?    | XX工作室 |
| 2D游戏   | 视角锁定   | 不支持  | 2D位图和8向位图动画                | 2D位图<br>(Stage3D加速，60帧频) | XX乾坤 | XX工作室 |
|        |        |      | 2D位图和8向位图动画<br>(Stage3D加速) | 2D位图                     |      |       |




# 从Flash 2D迈向3D会遇到的挑战



- 前端技术上的革命、更新换代，好多武功会被废掉：
  - 2D图形渲染技术：精灵、场景、特效、Avatar系统
  - 物理引擎，碰撞检测
  - 寻路、AI
- 可被延续的武功：
  - UI（暂时可重用，最终还是会被GPU based UI废掉）
  - 页游开发经验
  - AS3编程经验

# 从Flash 2D迈向3D会遇到的挑战 - Cont. 1



- 对开发者提出了更高的要求:
  - 3D游戏开发所需要的知识点与经验远高于2D开发
  - 3D游戏开发涉及一堆的技术知识点 →  90 3D技术知识
    - 3D数学、3D图形渲染、物理、AI寻路、场景组织管理
    - 对资源管理和加载器的设计和实现要求更高
    - 游戏性能优化方面，需要相当的经验 and 技能
  - 引擎研发人员：需要学会使用3dsMax
- 美术制作上也要求有更多的制作经验



# 3D技术学习步骤



## 学理论、打基础阶段

- 沉下心、踏踏实实打好基础，深入去搞清楚代码背后的数学基础、工作原理。
- 多看书：3D技术知识是通用的，直接去阅读Direct3D和OpenGL的书籍。Stage3D还没有好书。
- 多问问谷歌：网上有大量的Stage3D文章，但良莠不齐；最好是直接去看几个著名的老外的博客，不要看那些翻译版。

## 练习阶段

- 先不要使用高层引擎，自己写些代码、或改改现成的例子代码去验证、实践学到的理论知识。

## 使用引擎阶段

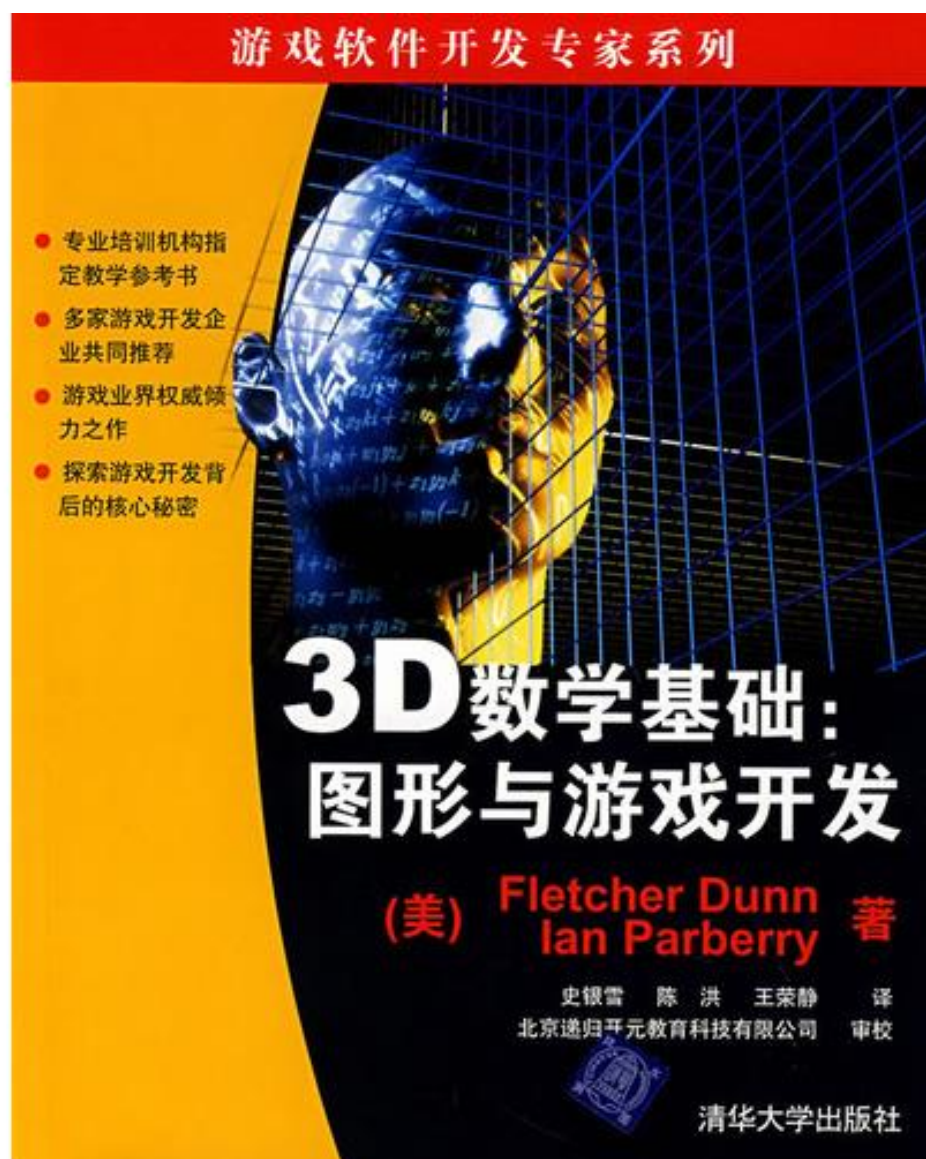
- 研究开源引擎源代码，站在别人的肩膀上，快速提升自己的功力。
- 最好能多研究几套引擎，博揽众长，最终揉合到你自己的引擎里。
- 善用反向工程！

# 学习资源推荐 — 书籍



## ● 三本经典书：

### 3D数学



### 3D图形学



### 3D游戏引擎设计



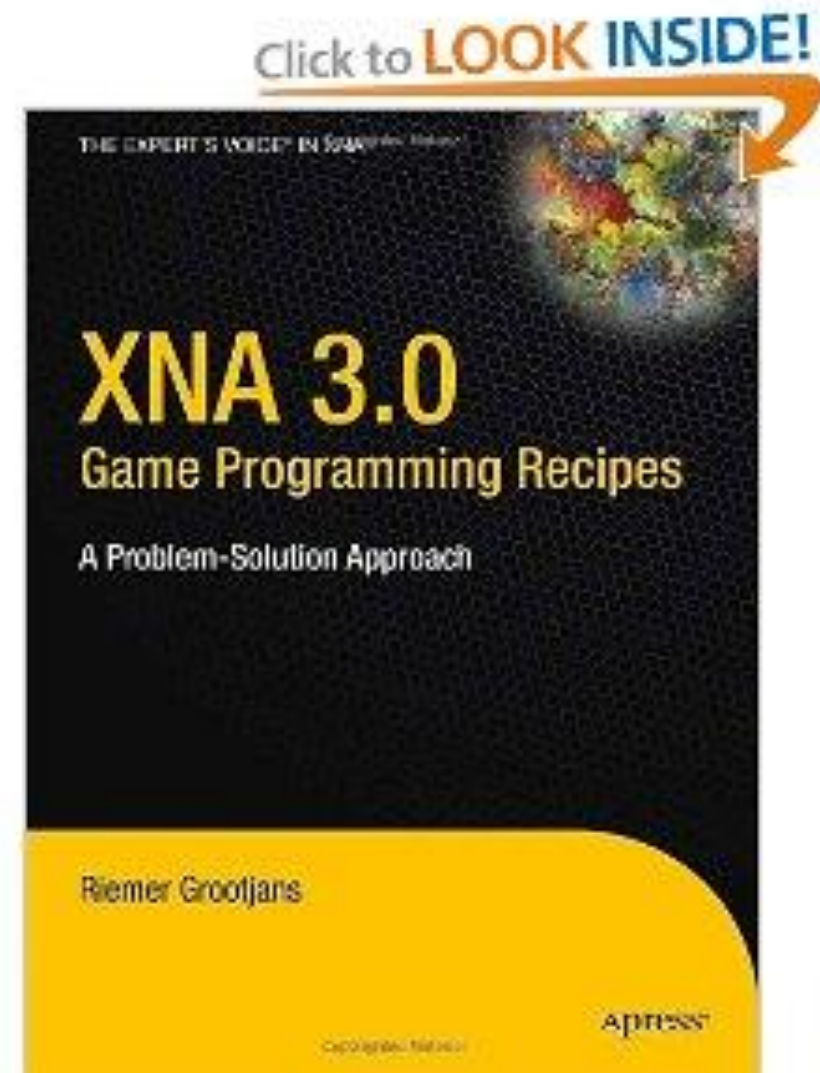
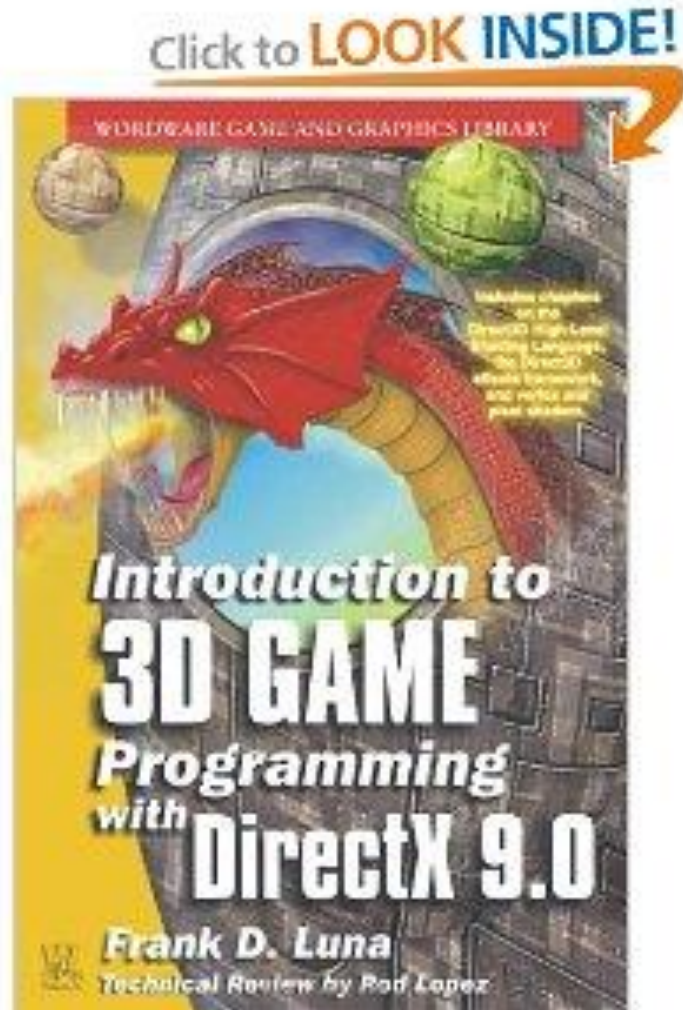


# 学习资源推荐 — 书籍



## ● DirectX和XNA领域实战型的书籍：

- Introduction to 3D Game Programming with DirectX 9, **Frank D. Luna**
- XNA 3.0 Game Programming Recipes: A Problem-Solution Approach
  - 该书的思路“提出问题→解决方案→工作原理→具体实现”





# 学习资源推荐 — 书籍



## ● 3D动画:

- Skinned Mesh Character Animation with Direct3D 9.0c , **Frank D. Luna**
- Character Animation With Direct3D



# 学习资源推荐 – 书籍



- 深入学习Shader编程的经典图书:

- GPU Gems 1和2: 有对应的中文版

- GPU精粹1: 实时图形编程的技术、技巧和技艺

- 译者: 姚勇、王小琴

- 出版日期: 2006年1月

- 出版社: 人民邮电出版社

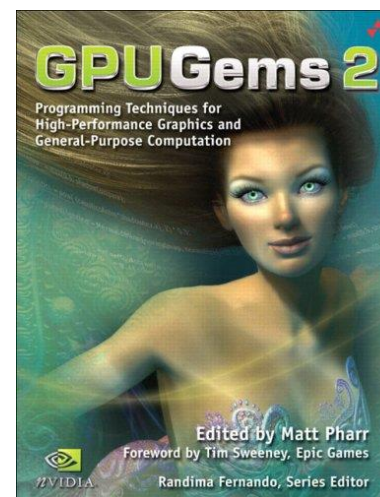
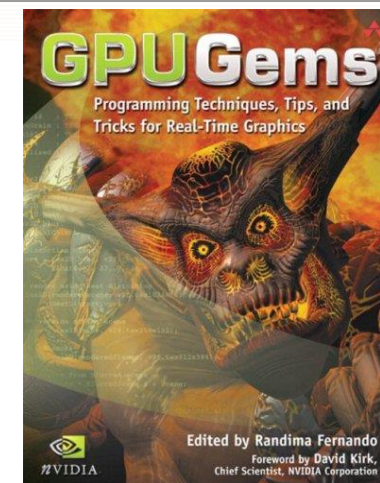
- GPU精粹2: 高性能图形芯片和通用计算编程技巧

- 译者: 龚敏敏

- 出版日期: 2007年5月

- 出版社: 清华大学出版社

- GPU Gems 3英文版



# 学习资源推荐 — 书籍



● Flash Stage3D: 唯一的一本Stage3D书，但写得一般般。

● Adobe Flash 11 Stage3D (Molehill) Game Programming Beginner's Guide





# 学习资源推荐 – 网上文章



- Stage3D & AGAL:
  - Adobe官方的Flash Stage3D教程： 直接看英文版的
  - Norbz's Dev Blog（需要翻墙）： Stage3D/AGAL from scratch;  
<http://blog.norbz.net>
  - Pierre's Blog: AS3 Level-4 Series Tutorial;  
<http://pierrechamberlain.ca/blog/tag/stage3d/>
- Away3D:
  - Away3D官网的教程
  - Jason Sturge's Blog: 不少Away3D的技术文章;  
<http://www.jasonsturges.com/tag/away3d/>
- Flash Player 11.x和Stage3D的最新特性:
  - ByteArray: Flash Player 11.x和Stage3D最新特性的试验性文章;  
<http://www.bytearray.org>



# 技术篇

## GPU和Shader Model

3D数学

3D变换

3D图形渲染－渲染管线

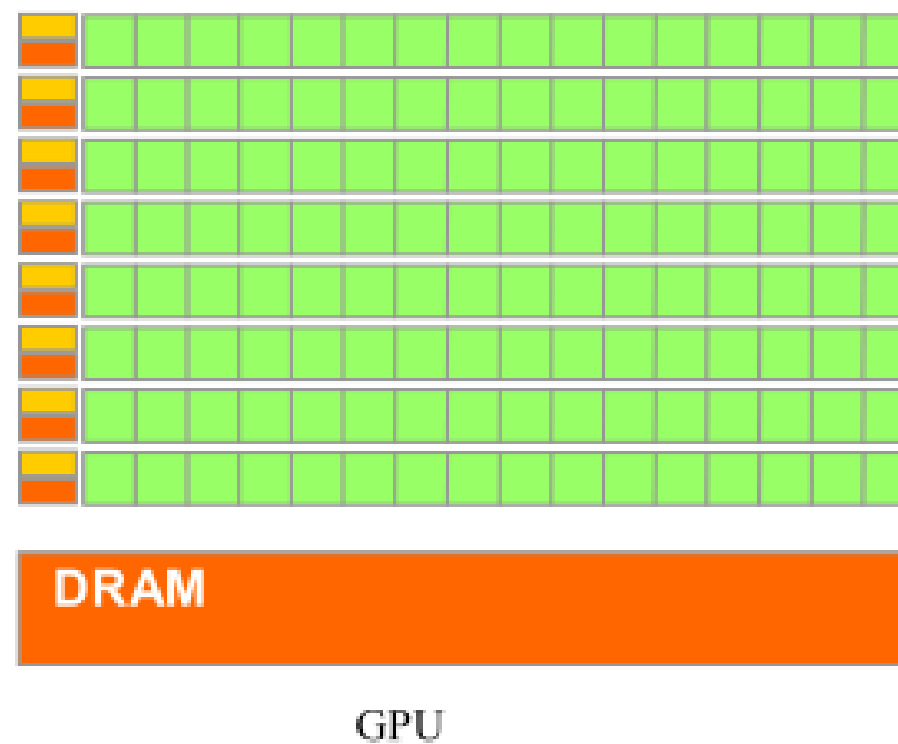
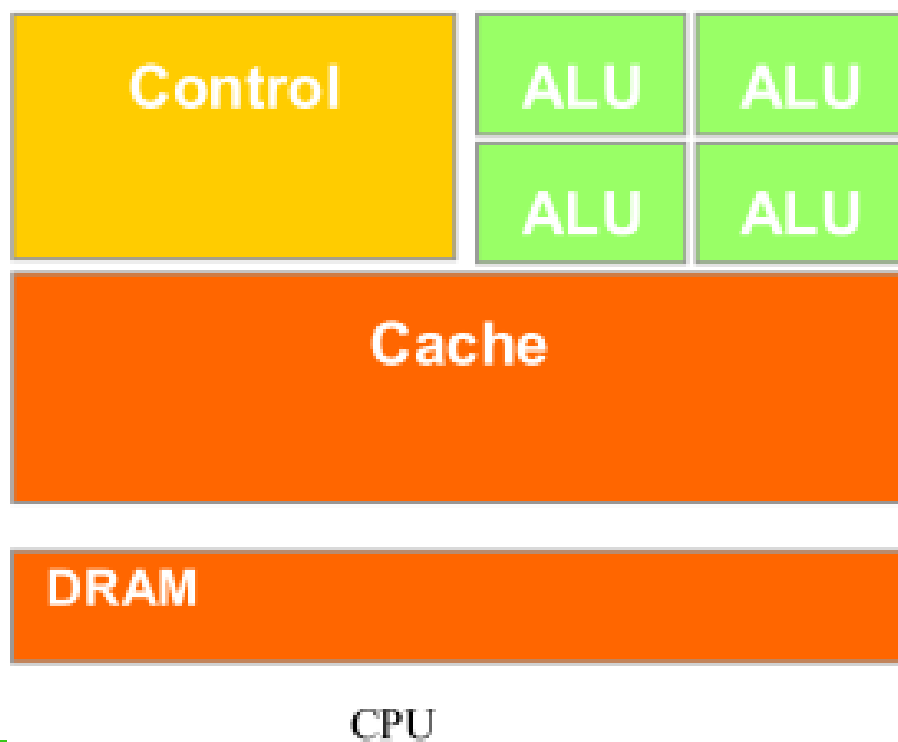
3D图形渲染－纹理映射

## 3D模型格式和3D动画

# GPU vs. CPU

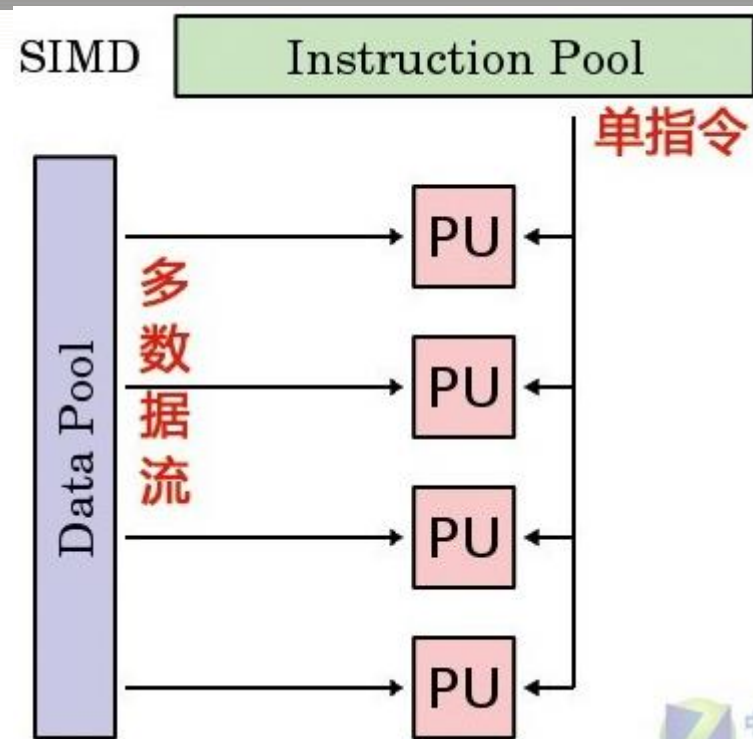


- **GPU具有高并行结构 (highly parallel structure) :**
  - CPU大部分面积为控制器和寄存器，一个时刻只处理一个数据，非真正意义上的并行计算。
  - GPU 拥有更多的 **ALU (Arithmetic Logic Unit, 逻辑运算单元)** 用于数据处理，而非数据高速缓存和流控制，这样的结构适合对密集型数据进行**并行处理**。





# GPU的SIMD&MIMD流式并行计算模式



```
for (int j = 1; j < height - 1; ++j)
{
    for (int i = 1; i < width - 1; ++i)
    {
        // get velocity at this cell
        Vec2f v = grid(x, y);

        // trace backwards along velocity field
        float x = (i - (v.x * timestep / dx));
        float y = (j - (v.y * timestep / dy));

        grid(x,y) = grid.bilerp(x, y);
    }
}
```

**C++**

```
void advect(float2 uv : WPOS,
           out float4 xNew : COLOR,

           uniform float dt, // timestep
           uniform float dx, // grid scale
           uniform samplerRECT u, // velocity
           uniform samplerRECT x) // state
{
    // trace backwards along velocity field
    float2 pos = uv - dt * f2texRECT(u, uv) / dx;

    xNew = f4texRECTbilerp(x, pos);
}
```

**Cg**

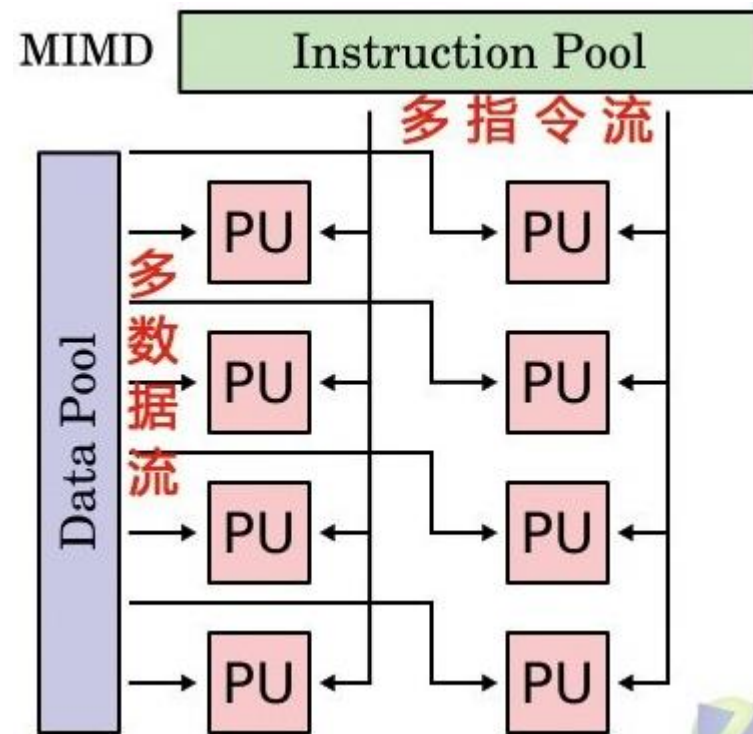


图 2 CPU 和 GPU 上的代码比较

# Shader Model



● Shader Model(SM): 可编程Shader的规格版本号。

- 程序长度限制
- 指令总类的增加
- 寄存器数量限制

| Shader Model | Direct3D   | OpenGL | Flash Stage3D                           | Video Card Example                                  |
|--------------|------------|--------|---|---|
| 1            | 8          |        |   | NVIDIA GeForce 3<br>ATI Radeon 8500                 |
| 2            | 9.0, 2002  | 2.x    | AGAL 1.0 Baseline,<br>AGAL 2.0 Extended | NVIDIA GeForce FX<br>ATI Radeon 9500/9600/9700/9800 |
| 3            | 9.0c, 2004 | 2.x    |   | NVIDIA GeForce 6800<br>ATI Radeon X800              |
| 4            | 10.x, 2007 | 3.x    |   | NVIDIA GeForce 8800<br>ATI Radeon HD 2900           |
| 5            | 11.x       | 4.x    |   | NVIDIA GeForce GTX 480<br>ATI Radeon HD 5870        |

# Shader Model Capabilities



|                           | SM 2.0/2.X            | SM 3.0        | SM 4.0             |
|---------------------------|-----------------------|---------------|--------------------|
| Introduced                | DX 9.0, 2002          | DX 9.0c, 2004 | DX 10, 2007        |
| VS Instruction Slots      | 256                   | $\geq 512^a$  | 4096               |
| VS Max. Steps Executed    | 65536                 | 65536         | $\infty$           |
| PS Instruction Slots      | $\geq 96^b$           | $\geq 512^a$  | $\geq 65536^a$     |
| PS Max. Steps Executed    | $\geq 96^b$           | 65536         | $\infty$           |
| Temp. Registers           | $\geq 12^a$           | 32            | 4096               |
| VS Constant Registers     | $\geq 256^a$          | $\geq 256^a$  | $14 \times 4096^c$ |
| PS Constant Registers     | 32                    | 224           | $14 \times 4096^c$ |
| Flow Control, Predication | Optional <sup>d</sup> | Yes           | Yes                |
| VS Textures               | None                  | $4^e$         | $128 \times 512^f$ |
| PS Textures               | 16                    | 16            | $128 \times 512^f$ |
| Integer Support           | No                    | No            | Yes                |
| VS Input Registers        | 16                    | 16            | 16                 |
| Interpolator Registers    | $8^g$                 | 10            | $16/32^h$          |
| PS Output Registers       | 4                     | 4             | 8                  |



# Stage3D AGAL Capabilities



- AGAL 1.0 Baseline: Flash Player  $\leq 11.5$
- AGAL 2.0 Extended: Flash Player  $\geq 11.6$

| Type                      | AGAL 1.0 Baseline | AGAL 2.0 Extended |
|---------------------------|-------------------|-------------------|
| Vertex Stream             | 8                 | 8                 |
| Vertex shader constants   | 128               | 250               |
| Fragment shader constants | 28                | 64                |
| Temporaries               | 8                 | 26                |
| Varying                   | 8                 | 10                |
| Texture sampler           | 8                 | 16                |
| Output colors             | 1                 | 4                 |
| Depth output              | N/A               | 1                 |
| Token count               | 200               | 1024              |

不再纠结骨骼动画模型里骨骼数量超标了

实现Avatar时不再纠结fs寄存器不够用了

AGAL代码的长度增加了，可以写更复杂的Shader程序了

# flash.display3d.Context3D的资源限制



## ● AGAL 1.0资源限制:

| 资源             | 允许的数量 | 总内存                 |
|----------------|-------|---------------------|
| Vertex buffers | 4096  | 256 MB              |
| Index buffers  | 4096  | 128 MB              |
| Programs       | 4096  | 16 MB               |
| Textures       | 4096  | 128 MB <sup>1</sup> |
| Cube textures  | 4096  | 256 MB              |

- AGAL 1.0 Program限制: 每个Program 200 个 opcode。
- 绘制调用限制: 每个 present()调用32,768个drawTriangles()调用。

① 350 MB 是纹理的绝对限制, 包括 mipmapping 所需的纹理内存。但是, 许多设备不支持这么多的纹理内存。为了最大限度地保证兼容性, 请将使用的纹理内存限制为 128 MB 或更少。

② 尽可能去使用Adobe的ATF压缩纹理来提高运行效率和节省纹理内存资源。



# 技术篇

GPU和Shader Model

3D模型格式和3D动画

3D数学

3D变换

3D图形渲染－渲染管线

3D图形渲染－纹理映射





# 技术篇

GPU和Shader Model

3D模型格式和3D动画

3D数学

3D变换

3D图形渲染－渲染管线

3D图形渲染－纹理映射



# 技术篇

GPU和Shader Model

3D模型格式和3D动画

3D数学

3D变换

**3D图形渲染－渲染管线**

3D图形渲染－纹理映射



# 技术篇

GPU和Shader Model

3D模型格式和3D动画

3D数学

3D变换

3D图形渲染－渲染管线

3D图形渲染－纹理映射





# 技术篇

GPU和Shader Model

3D模型格式和3D动画

3D数学

3D变换

3D图形渲染－渲染管线

3D图形渲染－纹理映射



# 引擎和工具篇

# Flash 3D Engines



| Engine               | Licence/Price               | Remark  |
|----------------------|-----------------------------|---|
| Adobe的亲生子女           |                             |   |
| Proscenium           | Free, Not Open source       | Adobe Lab里的实验性3D框架  |
| 老牌的三剑客               |                             |   |
| Alternativa3D        | Licence → Now Open source   | 日本Square Enix(史克威尔艾尼克斯)的Legend World用了它。  |
| Papervision3D        | Open source, free           | 在Stage3D时代，没跟上已被淘汰  |
| Away3D 4.0 gold, 4.1 | Open source, free           | Adobe的干儿子，现在有众多的粉丝。<br>特点：有老大Adobe投资；社区活跃<br>缺点：大而全，性能不好；版本间API兼容性<br>通常被大家当作入门学习的引擎，和最终自研引擎和工具的库来使用。 |
| 后起之秀                 |                             |   |
| Aerys Minko          | No licence, just consulting | 缺点：Pro版要收费；文档缺乏；版本间兼容问题<br>特点：设计思想前卫(插件式组装，不采用AS3 Event而是Signal同步函数回调，不采用重量级的MVC框架)                   |
| 收费的引擎                |                             |   |
| Flare3D              | Licence, price unknown      | 特色：3D工具和高层Shader语言FLSL  |



# 自研3D MMORPG引擎的目标

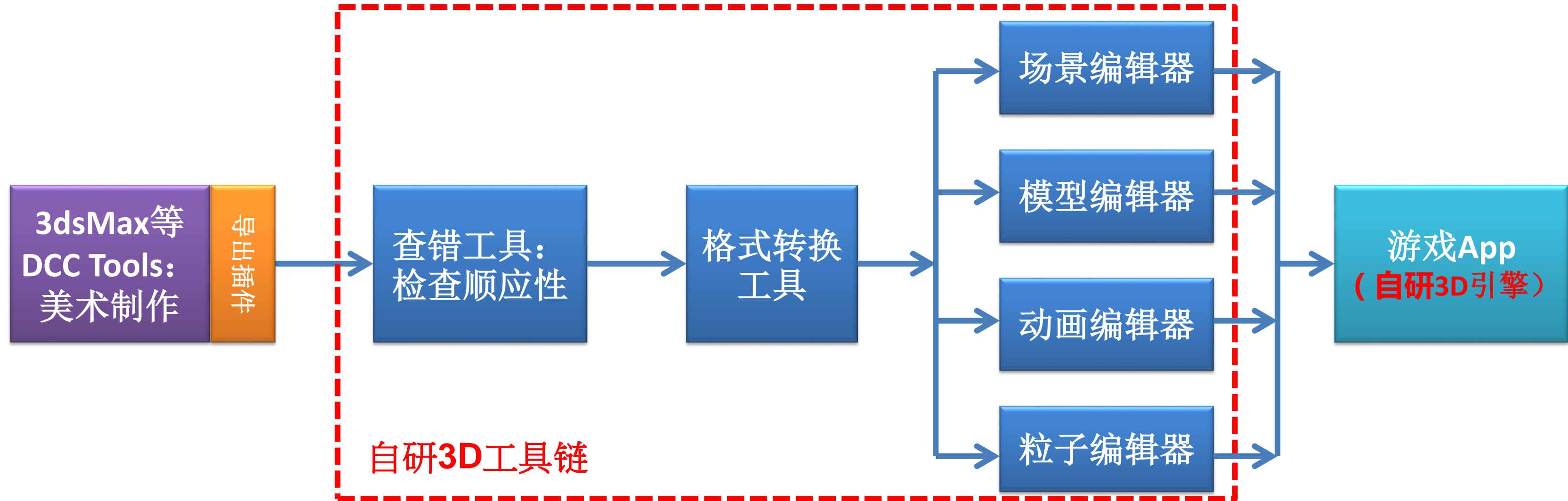


- 全3D游戏引擎：
  - 3D模型渲染场景、精灵、动画
  - 无锁定视角
  - 支持场景缩放
  - 支持鼠标锁定（FPS游戏需要）
  - 60帧频的爽滑

● 能够向下灵活支持如下几种产品类型：

| 产品类型   | 视角     | 场景缩放 | 精灵渲染方式   | 场景渲染方式  |
|--------|--------|------|--|---|
| 全3D游戏  | 全视角无锁定 | 支持   | 3D模型 + 3D动画                                      | 3D模型  |
| 3D游戏   | 视角锁定   | 不支持  | 3D模型 + 3D动画                                      | 3D模型  |
| 2.5D游戏 | 视角锁定   | 不支持  | 3D模型 + 3D动画                                      | 2D位图  |
| 2D游戏   | 视角锁定   | 不支持  | <del>2D位图和8向位图动画</del>                           | <del>2D位图</del><br><del>(Stage3D加速, 60帧频)</del> |
|        |        |      | <del>2D位图和8向位图动画</del><br><del>(Stage3D加速)</del> | <del>2D位图</del>                                 |

# 自研3D引擎工具链



自研3D引擎作为运行支撑环境

# 常用3rd DCC Tools介绍



- 3D建模和动画制作工具：
  - 3dsMax + 各种导入/出插件
  - Maya
- MD5模型格式的工具：
  - MilkShape3D（收费）：建模，动画
  - QuArk（开源）：场景编辑器
  - MD5 Model Viewer（开源）：场景和模型查看器
- Stage3D的工具
  - iFlash 3D Librarian（收费）：<http://iflash3dlibrarian.net/features/>
    - Just import one or more Collada files into iFlash3D Librarian, and export it to a highly optimized and compressed library file, containing all your models and textures, that is ready to be used by Adobe Flash Stage3D.
  - Flare3D引擎里的工具（收费）
  - Minko3D引擎里的Minko Studio（收费）、Shader Lab（收费）
  - Away3D引擎里的一些工具（开源）



# 3<sup>rd</sup> DCC Tools的问题



## ● 3<sup>rd</sup> DCC Tools的问题:

- 1) 3dsMax插件导出的模型，有可能不吻合我们引擎的要求，例如，面数、骨骼数、材质组织规则、命名规则等。（这也就是为什么我们要制定3D模型制作规范的原因之一）
- 2) MD5那些工具主要给Quake4和Doom3游戏的MOD爱好者使用的。

# 自研3D工具链会遇到的问题



## ● 自研3D工具链会遇到的问题：

- 1) 3D工具链的开发工作量(端游经验值)：巨大 =  $2 * \text{引擎研发工作量}$
- 2) 持续性的工作：且后续在3D引擎完善和游戏研发中，会持续有新需求。