中国游戏开发者大会

CHINA GAME DEVELOPERS CONFERENCE

2013年7月26-28日

LET US SHARE

www.chinagdc.com.cn

让OpenGL成为你的竞争优势

John McDonald
高级软件工程师，NVIDIA

gameworks.nvidia.com

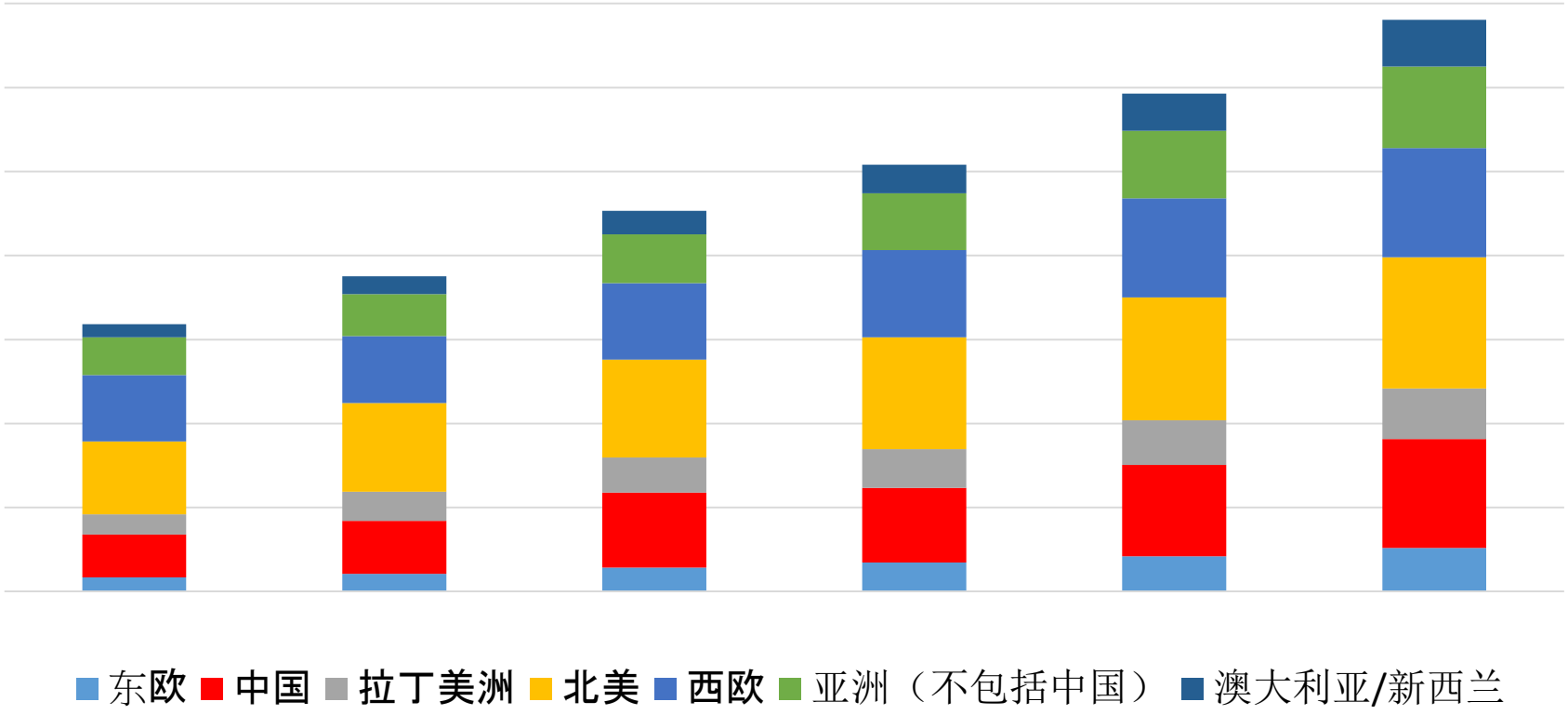
概述

- 市场概况
- 为什么要使用OpenGL
- 把Direct3D程序移植到OpenGL
- 移植到移动平台或者其他平台

市场概况

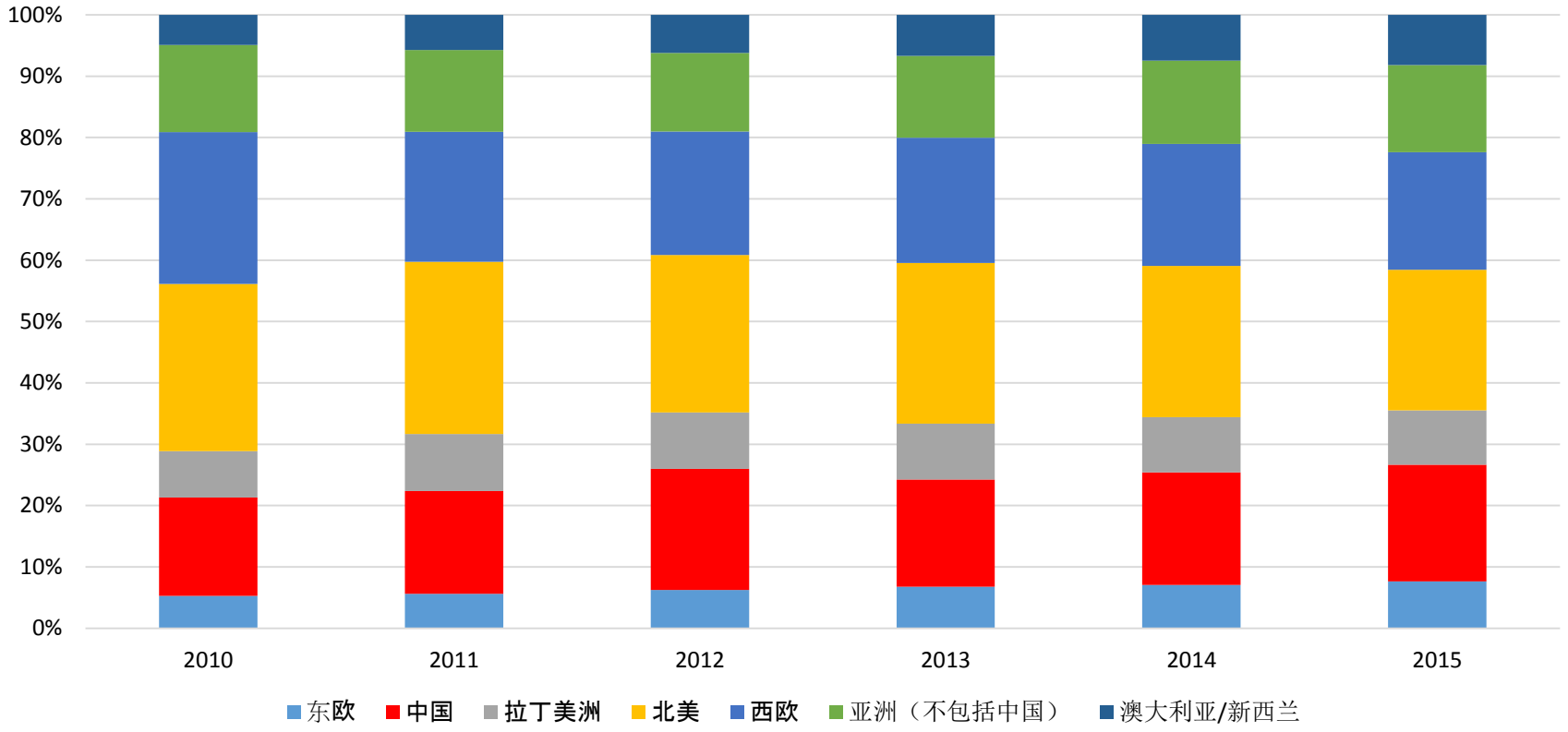
市场概述

按照区域划分的全球PC销量



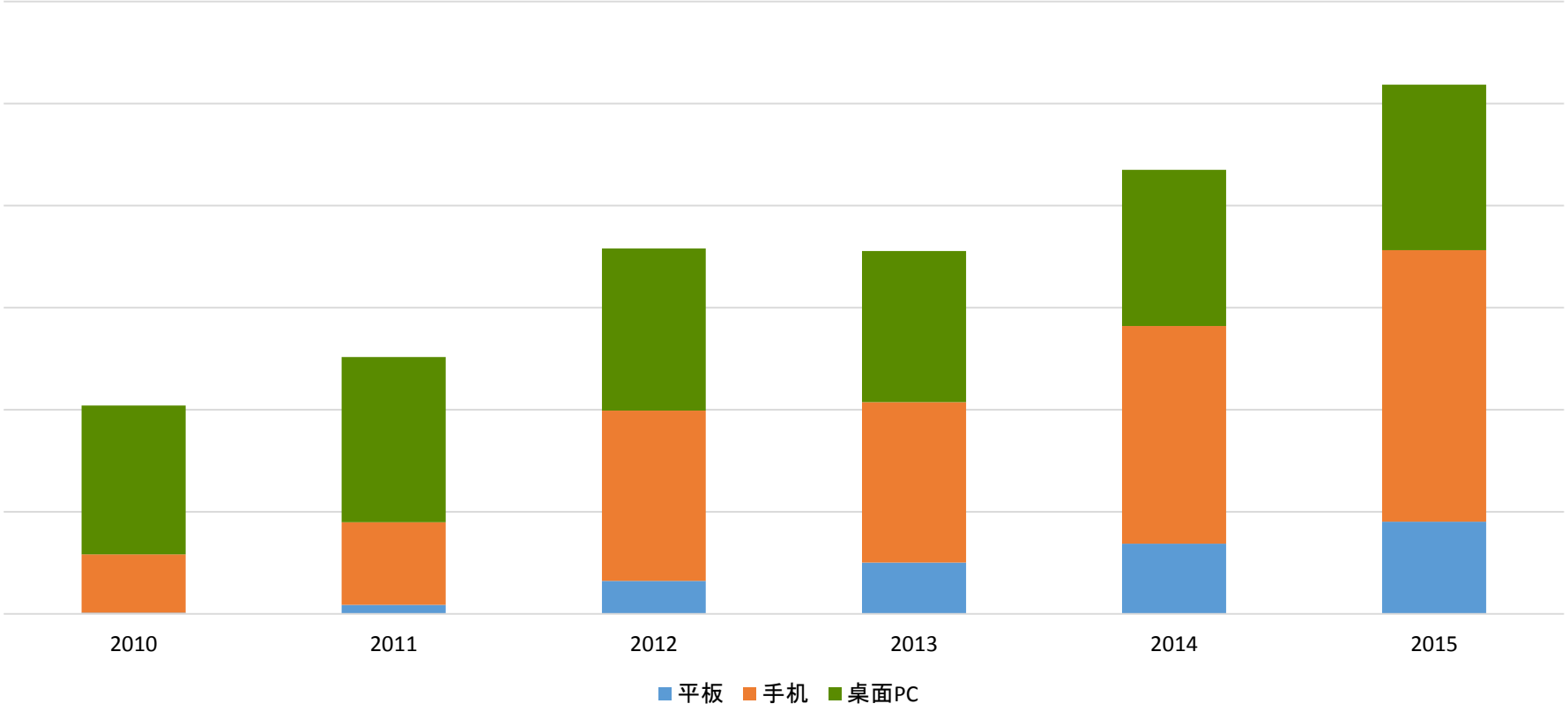
市场概述（续）

按照区域划分的PC销量比例



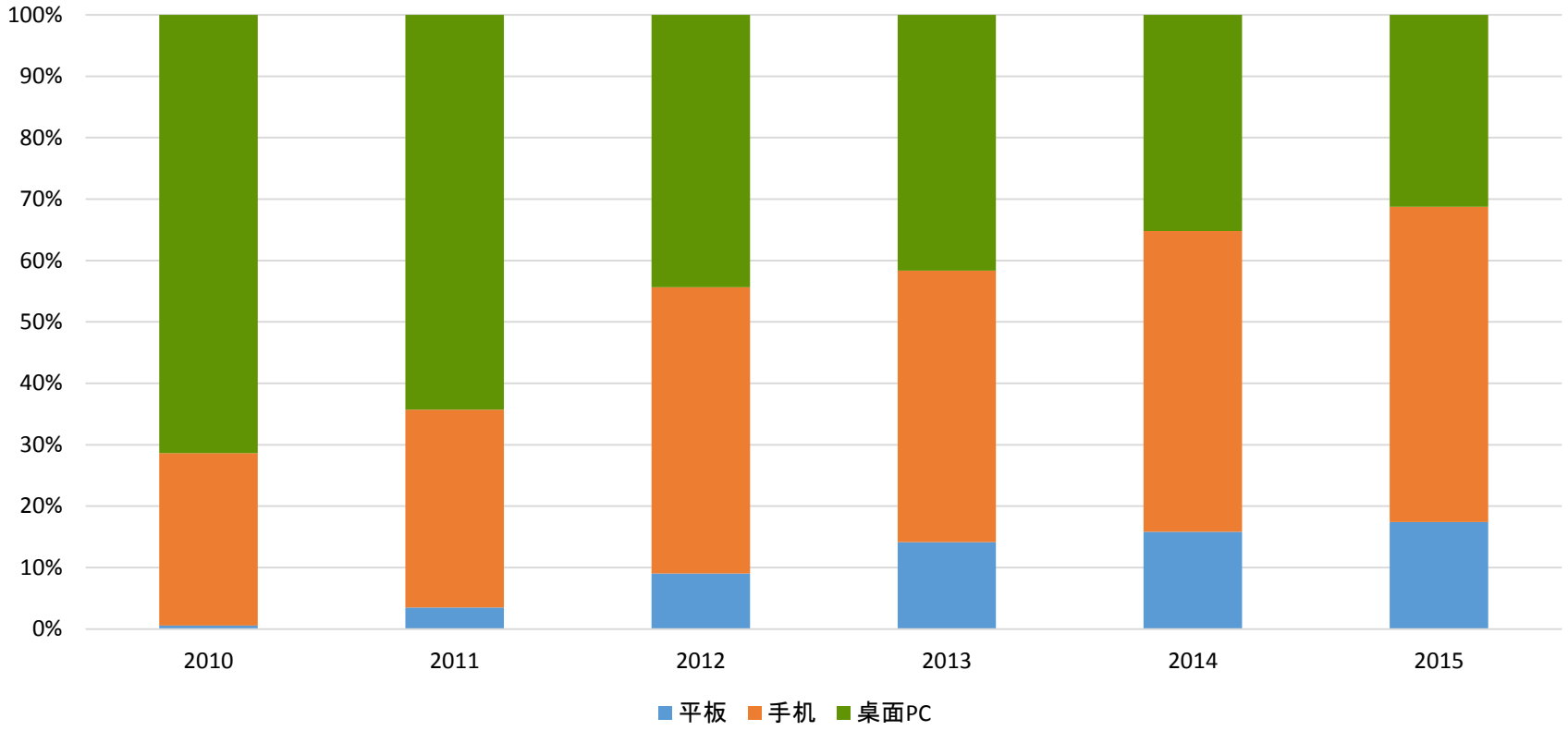
国内的市场划分

国内的PC市场



国内的市场划分（续）

国内的PC销量比例



为什么要使用OpenGL

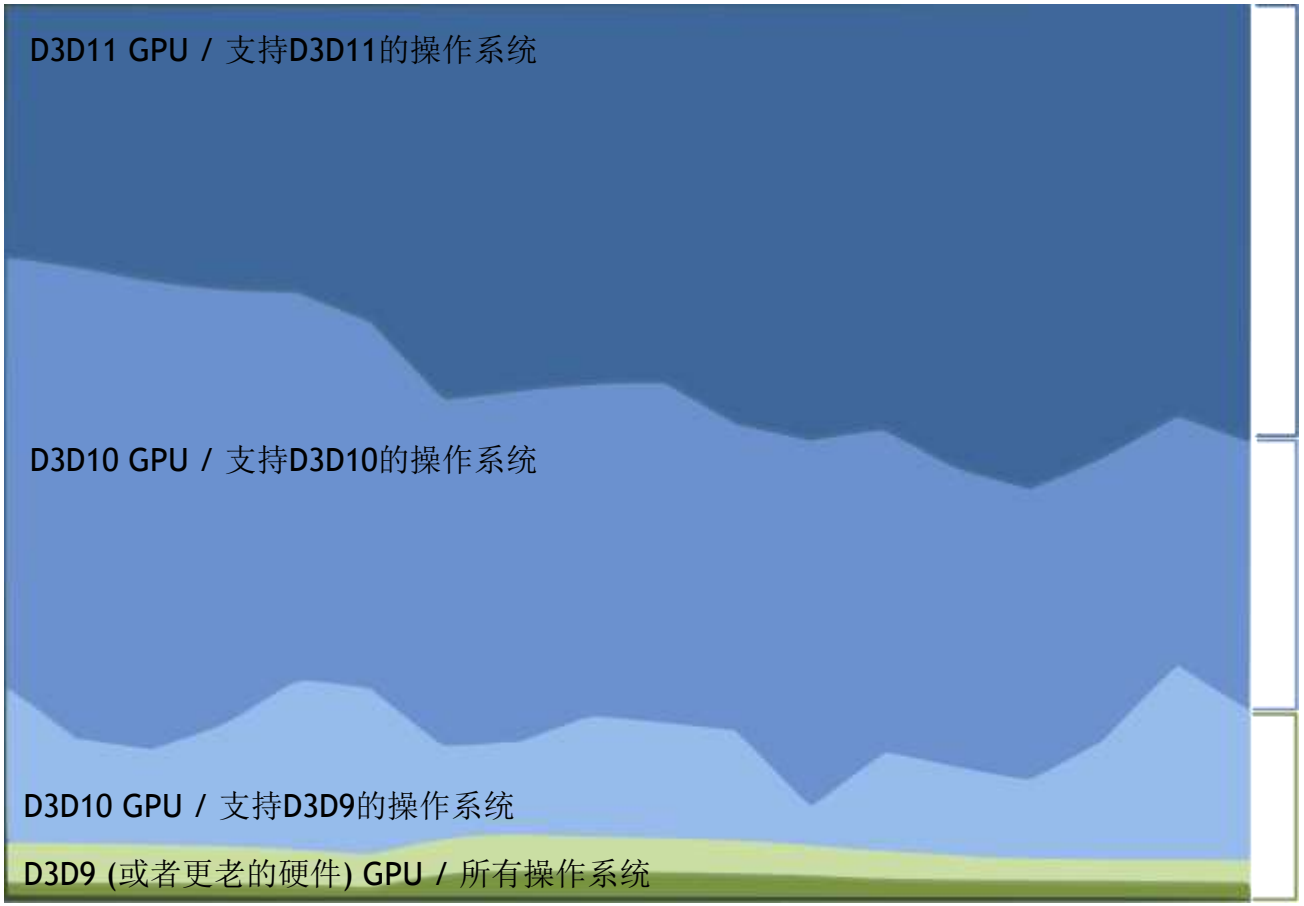
为什么要使用OpenGL

- OpenGL会根据硬件的能力提供功能，而不是操作系统
- 前面的表格中的所有设备都支持OpenGL的
- 而只有一部分设备（桌面和平板设备）支持Direct3D，而且其份额在不断缩小
- 中国有大量的GPU资源，不过大部分都还运行在xp上面
  - OpenGL可以为所有用户提供Dx10/11的特性

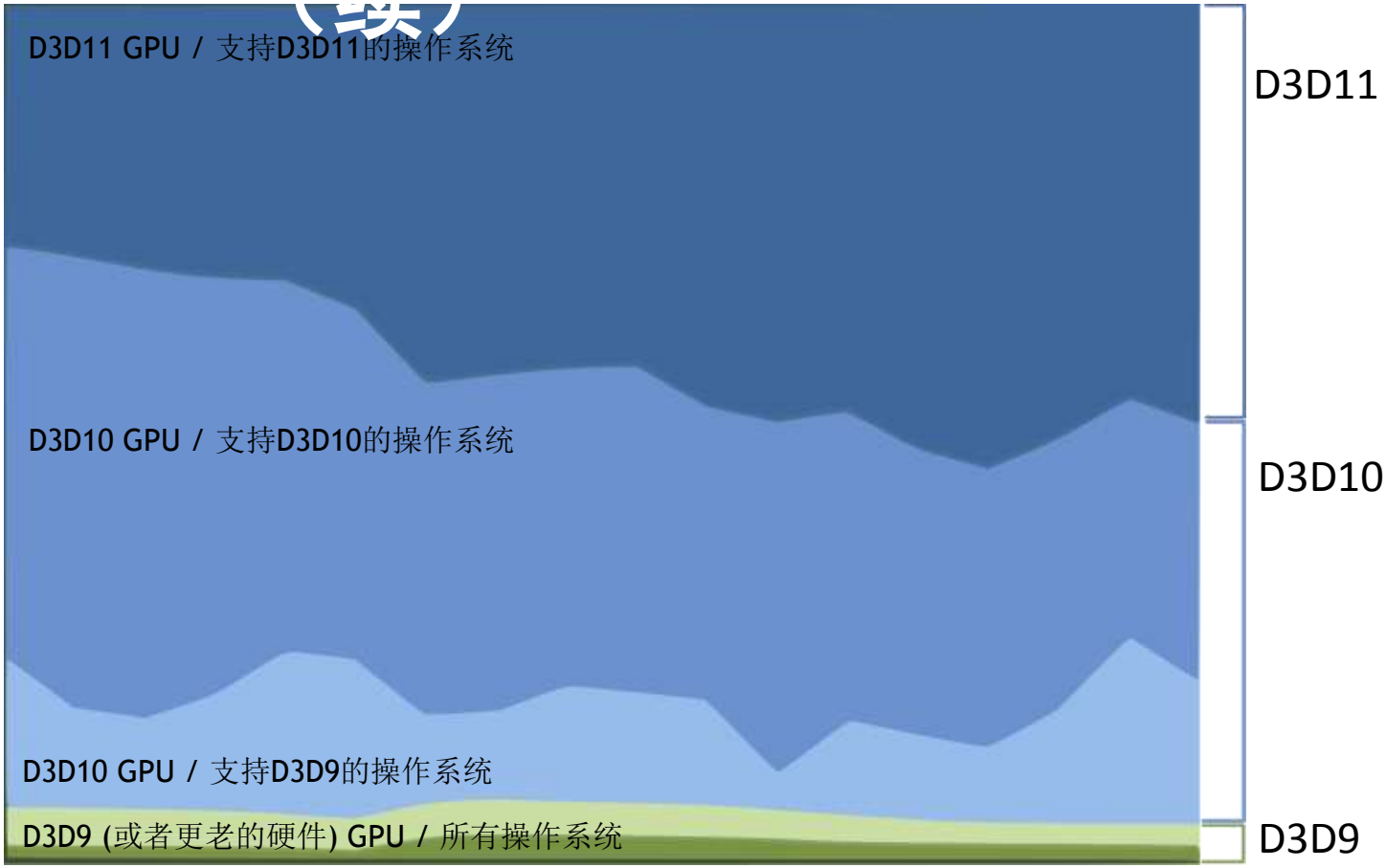
为什么要使用OpenGL（续）

- OpenGL的规范是公开的
- 对于所有有兴趣的人来说，都是可以直接获得OpenGL的（有些可能会需要一些资金，但是不是很多）
- 从一个供应商开始，OpenGL扩展的是非常快的
- OpenGL是非常强大的

Direct3D支持 – 桌面/平板



Direct3D支持 – 桌面/平板



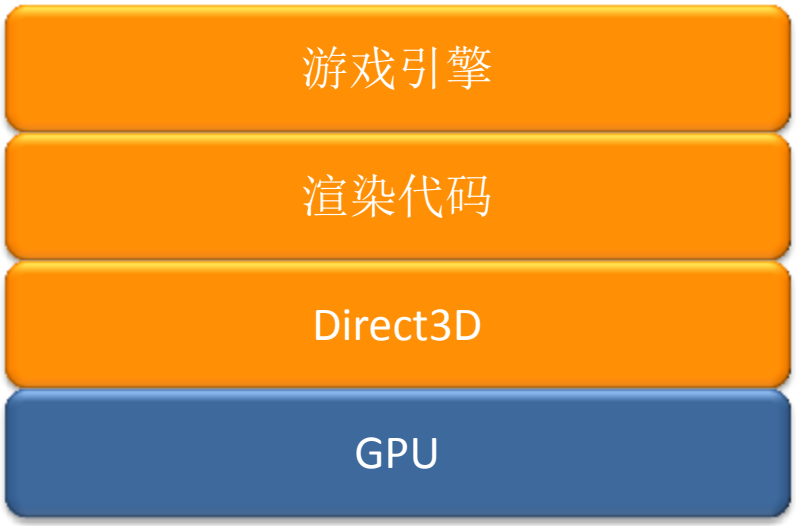
从Direct3D移植到OpenGL

你应该支持哪个版本的OpenGL？

- DX9 ≈ OpenGL 2
  - 可编程着色器
- DX10 ≈ OpenGL 3
  - 精简的API接口
  - 几何着色器
- DX11 ≈ OpenGL 4
  - 曲面细分和通用计算

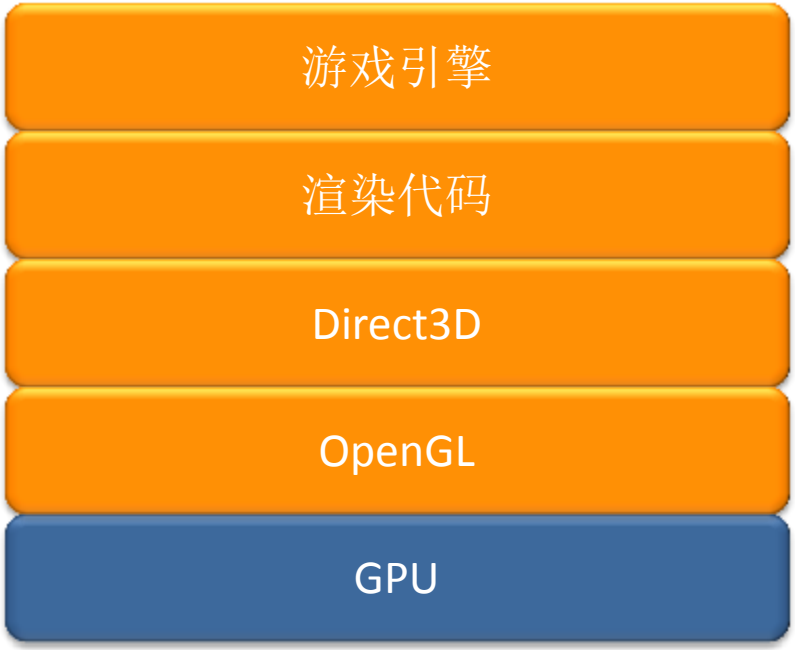
togl

- “移植到OpenGL”
- 用OpenGL实现DirectX9/10/11
- 最初由Valve实现
- 以Dll形式应用存在
- 大部分时候（99.9%），引擎代码是不需要关心具体API的种类的（即使是渲染部分的代码）



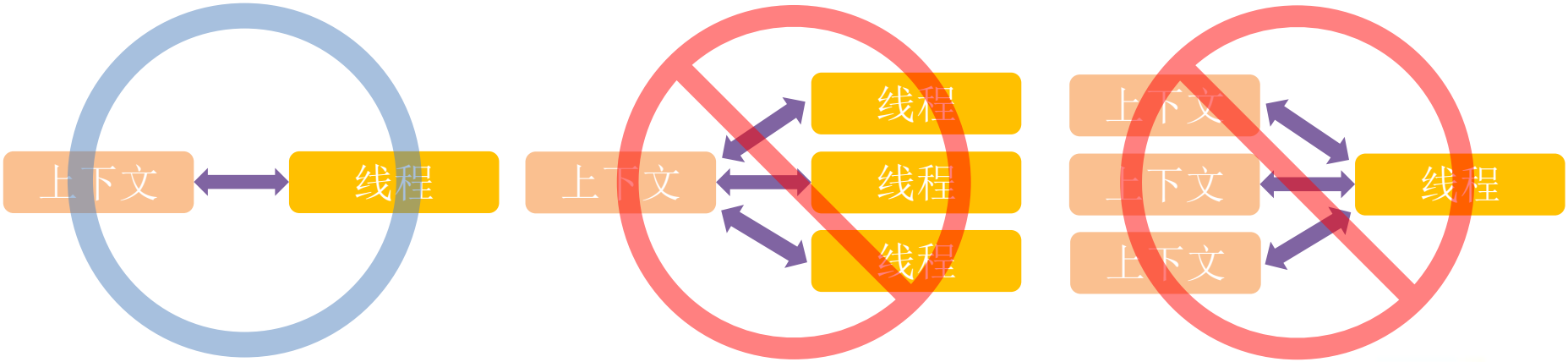
togl（续）

- “移植到OpenGL”
- 用OpenGL实现DirectX9/10/11
- 最初由Valve实现
- 以Dll形式应用存在
- 大部分时候（99.9%），引擎代码是不需要关心具体API的种类的（即使是渲染部分的代码）
- 性能需要特殊注意，但不是很大的问题 – 上面的实现要比之前的实现快20%



OpenGL和D3D的区别

- OpenGL拥有线程数据
  - 一个线程最多同时拥有一个上下文
  - 一个上下文同时最多存在于一个线程
  - 一个没有上下文的线程中调用的GL函数是没有作用的
  - MakeCurrent会影响当前线程与上下文之间的关系



OpenGL陷阱

- 渲染过程中存在一些“陷阱”
  - 功能方面
    - 贴图状态
    - 左右手坐标系
    - 像素中心的差别
  - 性能方面
    - MakeCurrent
    - 驱动串行化
- 不同供应商的实现 – 确认在不同的供应商的硬件上测试你的代码

贴图状态

- 默认情况下，OpenGL会在与贴图直接相关的一个表头中存储贴图的访问方式

* Not to scale

- 下面的代码并不能完成你想要的功能

贴图状态（续）

```
glBindMultiTextureEXT( GL_TEXTURE0 + 0, 7 );
glTexParameteri( GL_TEXTURE_2D,
                 GL_TEXTURE_MIN_FILTER,
                 GL_NEAREST );

glBindMultiTextureEXT( GL_TEXTURE0 + 1, 7 );
glTexParameteri( GL_TEXTURE_2D,
                 GL_TEXTURE_MIN_FILTER,
                 GL_LINEAR );
// Draw
```

ARB_sampler_objects

- 利用ARB_sampler_objects的扩展，贴图可以通过不同的单元用不同的方法进行访问
- 相对于贴图表头而言，采样器的优先级更高一些
- 如果采样器被设置成空，硬件会读取表头数据
- 不需要进行Shader编程
  - http://www.opengl.org/registry/specs/ARB/sampler_objects.txt

使用采样器

- ```
  Gluint samplers[2];
  glGenSamplers( 2, samplers );
  glSamplerParameteri( samplers[0],GL_TEXTURE_MIN_FILTER,
                       GL_NEAREST );
  glSamplerParameteri( samplers[1],GL_TEXTURE_MIN_FILTER,
                       GL_LINEAR );

  glBindSampler( 0, samplers[0] );
  glBindSampler( 1, samplers[1] );
  glBindMultiTextureEXT( GL_TEXTURE0 + 0, 7 );
  glBindMultiTextureEXT( GL_TEXTURE0 + 1, 7 );
  // Draw
  ```
  ```

其它GL与D3D的不同（续）

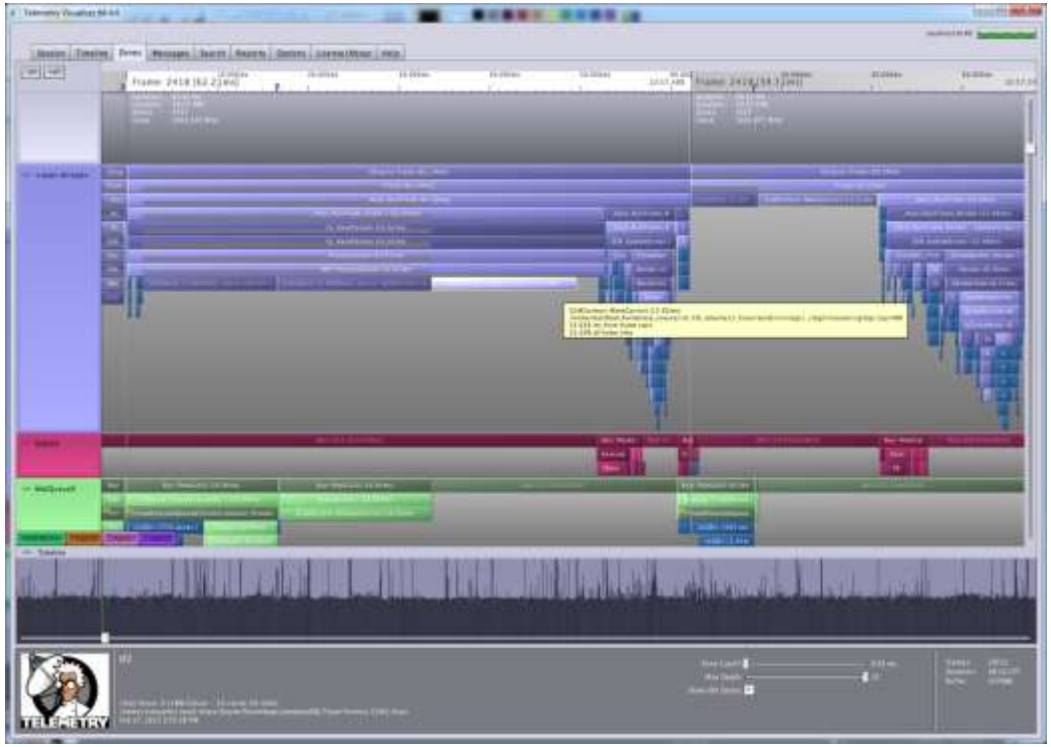
- 左右手坐标系
  - D3D 是左手系的，GL是右手系的
  - GL中贴图坐标原点在左下角（需要在v方向上翻转坐标）
  - 可以考虑倒着渲染，然后在左后做翻转
- GLSL 默认使用列主优先的矩阵
  - 包括在声明 constants/uniforms 的时候
- 像素中心
  - OpenGL与D3D10+ 相当

MakeCurrent 问题

- 在军团要塞2里面造成了几个bug
- 字体渲染的瑕疵（生成文字的线程试图更新贴图，但是没有得到上下文）

MakeCurrent的性能

- 在这里使用单线程是最好的
- MakeCurrent是十分昂贵的-尽量不要每帧都调用，即使每帧只调用一两次



驱动串行化

- 当前的OpenGL的驱动都是双核多线程化的
  - 你的应用和一个简单的驱动层交互
  - 这个简单的驱动层将数据传给另一个线程以准备提交
  - 与D3D相似
- 调用某些渲染函数会引起这个简单的驱动层清空所有的工作从而和主线程发生同步
- 这是十分昂贵的

已知的易引起串行化问题的函数

- glGet(…) – 大多数这样的函数会造成串行化；隐藏状态(就像D3D)
- glGetError – 使用了 ARB_debug_output!
- 有返回值的那些函数
- 那些不能够或者很难预先确定内存大小的拷贝函数

检测驱动串行化

- 使用ARB_debug_output!
- 在你的回调函数中设置断点，从调用堆栈中查找引起问题的函数调用
- 在ARB_debug_output消息中需要“同步调用：阻碍了多线程的优化”

Shader的转换

- 你有一大堆HLSL，你需要得到GL GLSL
  - ARB_vertex_program/ARB_fragment_program是一个可行的选择，但是仅限于DX9
    - 不存在ARB_tessellation_program

Shader的转换（续）

- 方法：编译HLSL，将byte code转换成类汇编的GLSL
- 优点：批量处理Shader
- 优点：制作迅速
- 缺点：难于调试及查找问题
- 缺点：生成的GLSL代码会有fxc的痕迹
- 缺点：调试起来很费力

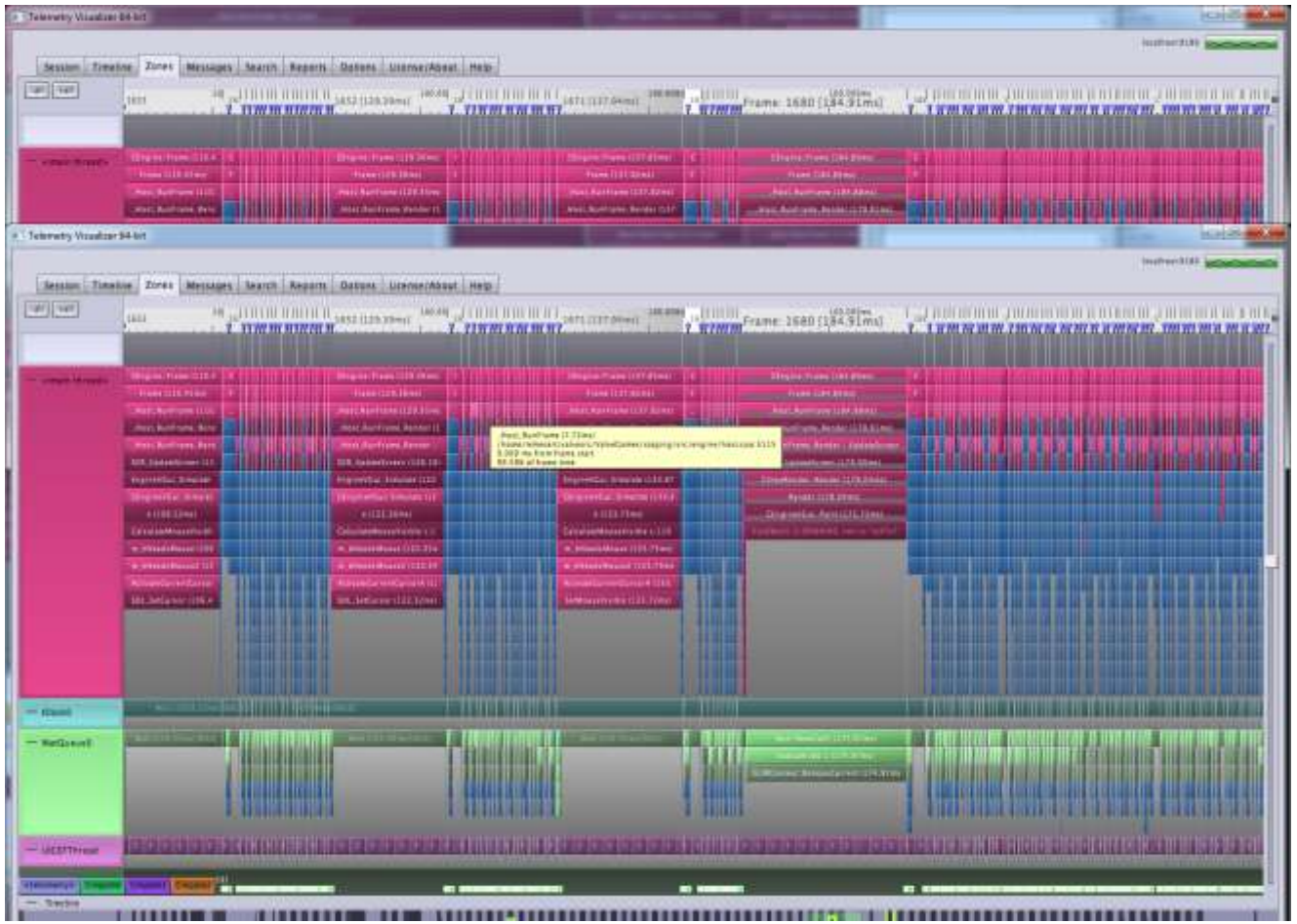
其他的转换方法

- 使用开源的工具
  - HLSLCrossCompiler – D3D11 only (SM4/5)
  - MojoShader – SM1/2/3
    - 在一些游戏和引擎中有附带，包括Unreal Tournament 3, Unity

- https://github.com/James-Jones/HLSLCrossCompiler
- http://icculus.org/mojoshader/

优化建议

- 性能调试
- 性能调试
- 性能调试



优化建议（续）

- 为了得到最高的性能，有时需要对不同硬件做针对性的优化
- 事实上，你可能已经做过类似工作了
- 现在硬件的行为已经有公开的规范了

GL调试与性能分析工具

- NVIDIA Nsight 支持 GL 4.2 Core
  - 以及一些扩展
  - 将支持更多的扩展/特性！
- PerfStudio 和 gDEBugger
- CodeXL
- Apitrace
  - 开源的API跟踪工具有一些扩展性问题，Valve正在修复

GL调试的技巧

- 比较D3D和GL的图片
- 让它们在相同的平台上都能工作
- 要点：
  - 让游戏在两台机器上运行，把输入同时发送到两台机器上，实时对比图像

移植到移动平台或其他平台

gameworks.nvidia.com

GL的组成

- OpenGL 4.x （桌面）
- OpenGL 2.1 （兼容苹果OSX）
- OpenGL 3.2 （OSX内核）
- OpenGL ES （移动平台）
- WebGL

Regal

- GitHub平台上，社区维护的项目
  - http://github.com/p3/regal
- 整合GL - 编写一个可移植的后端
- 软件兼容性
  - 如果驱动不支持
  - 立即模式，固定管线同样可以工作
  - 对于大量的图形程序而言，这种模式是比较推荐的
- 广泛支持可模拟的特性
  - DSA, VAO
  - 计划中：SSO, path rendering, enhanced display lists

Regal（续）

- 生态环境的基础
  - 用于调试的集成化http服务器
    - 监听或者修改上下文状态或对象，暂停渲染
  - API dump 日志
  - 集成了Apitrace
  - Shader,贴图dump和置换
- 开源 - BSD 认证
  - Githud（ http://github.com/p3/regal ）
  - 来自各地的诸多贡献者
- 平台支持
  - Windows, OS X, Linux, Android, iOS, NaCl

提问？

- Jmcdonald at nvidia dot com
- 关于D3D转换到GL的具体细节，请查询“Porting Source to Linux: Valve’s Lessons Learned”

附录

- 一些其他的OpenGL资源

有用的扩展

- EXT_direct_state_access
- EXT_swap_interval (和 EXT_swap_control_tear)
- ARB_debug_output
- ARB_texture_storage
- ARB_sampler_objects


LET US SHARE
CGDC 2013