



中国游戏开发者大会

CHINA GAME DEVELOPERS CONFERENCE

为安卓智能电视开发和移植PC画面级 别的游戏

Rev Lebaredian, 游戏内容部门高级工程总监

内容提要

- 把PC或者AAA级主机游戏移植到AndroidTV上有哪些必要步骤？
 - 游戏成功运行
 - 游戏的运行性能

- 我们希望SHIELD/Android成为大家最易上手的移植平台！



《无主之地 2》 《无主之地：前传》



《无主之地2》 《无主之地：前传》

- 由Gearbox和2K Games开发
- 无主之地2 (BL2), 于2012年9月发售
- 无主之地：前传(TPS), 于2014年10月发售
- 基于订制的虚幻引擎3 (2011年6月版)
- 发售平台为Windows, X360, PS3, Linux, Mac
- PC渲染引擎基于D3D9
 - 后来为了Linux的移植加入了OpenGL的支持(TPS 在发售时就有GL支持)



NVIDIA SHIELD



虚幻引擎3 和安卓

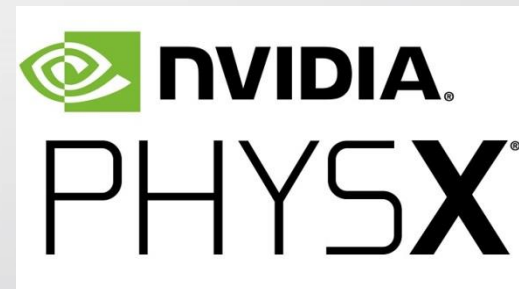
- 无主之地的UE3引擎对安卓平台的支持...
 - UE3引擎基本可以在安卓上跑起来
 - 第三方库文件
 - OpenGL ES2 移动渲染引擎
 - 一些例子代码
- 我们可以使用的部分...
- 我们需要加入/替换的功能...



第三方库

- UE3引擎版本越新，安卓支持越好
- 对于我们来说(2011年6月版)
 - Google protobuf
 - AkAudio/Wwise
 - PhysX
 - Bink
 - FaceFX
 - Scaleform!!!
- 移植/集成的难度各有不同

Wwise®
empowers audio creators



渲染引擎

- 已有的渲染引擎可选项

- D3D9 API
- OpenGL 3.2 PC端API
- OpenGL ES2 API

- 增加的: OpenGL ES3.1 API

	PC	SHIELD	Other Android
D3D9	YES	NO	NO
GL 3.2	YES	YES	NO
ES 2.0	YES*	YES	YES
ES 3.1	YES*	YES	YES

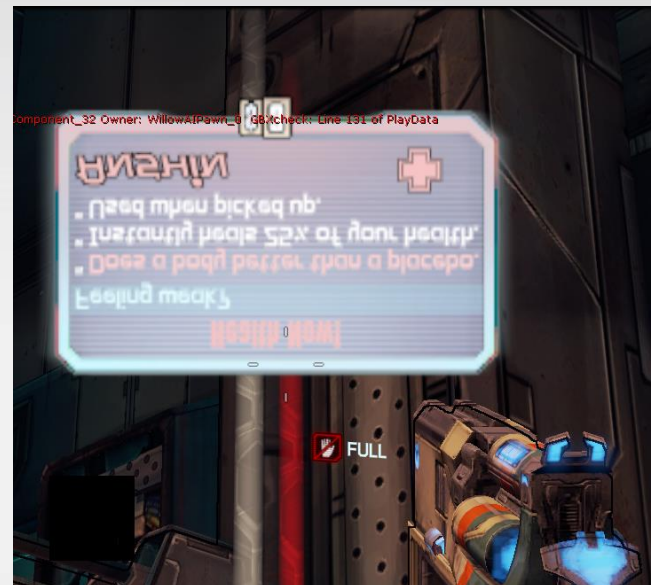
PC端OpenGL的功能

- 你有游戏已经支持OpenGL了？ 那你赚大发了！
- 不然的话.....

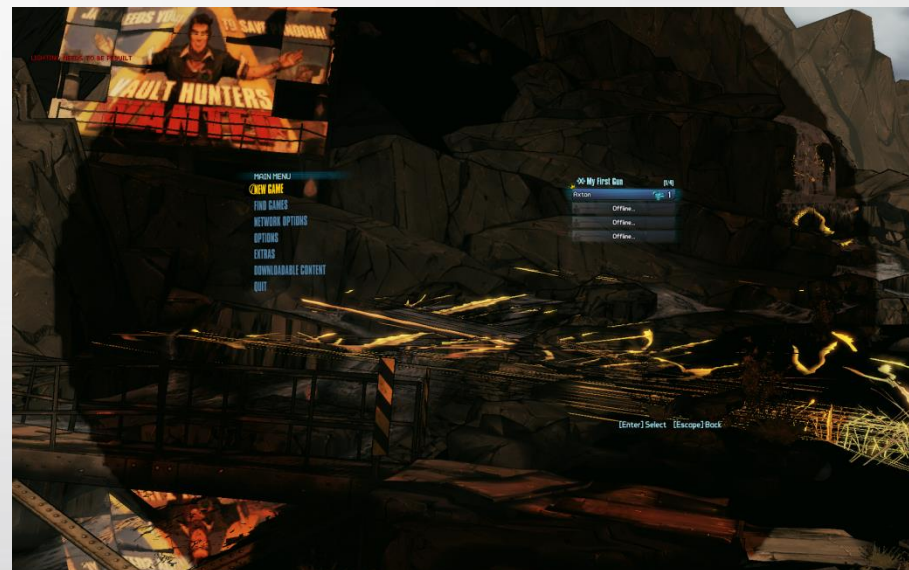


OpenGL注意事项

- 影像倒置！渲染瑕疵！
- 或者更坏的情况，渲染不出来了
- UE3已有的渲染效果
- 开发商自己写的渲染效果
- 一些常见的问题（GL vs D3D）
 - 纹理空间的倒置 ($t = 1.0 - t$)
 - 静态纹理 vs 动态生成纹理 vs 渲染Buffer纹理
 - 剪裁空间Z的不同 ($z = 2z - 1.0$)
 - NaN/inf 的处理 (abs/epsilon)
- NSight OpenGL 工具可以极大的帮助你调试



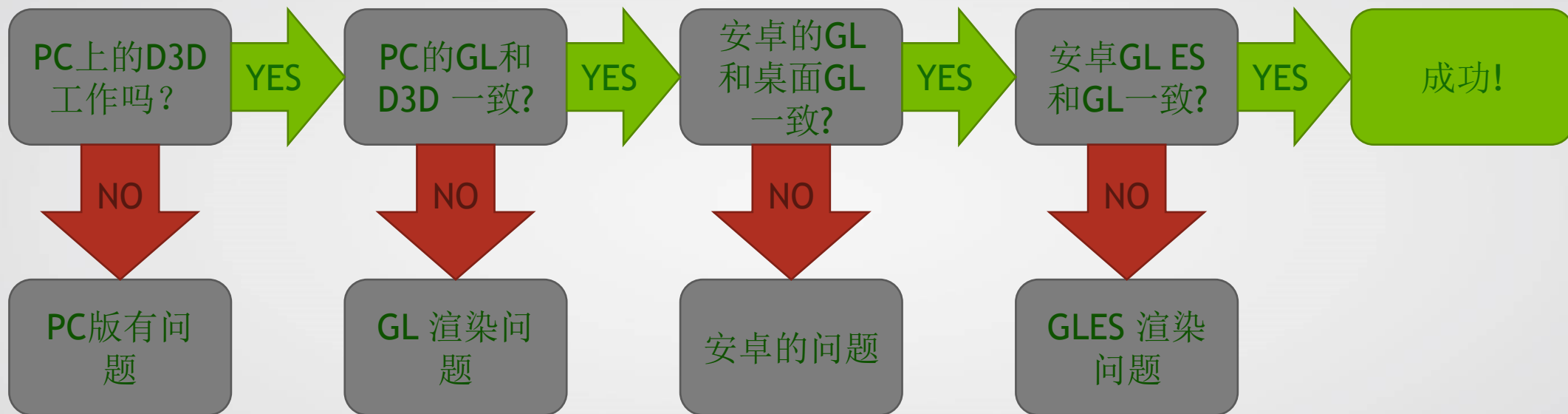
更多的Bug例子



安卓OpenGL的功能

- UE3自有的安卓支持很有限
 - OpenGL ES2渲染器只支持一些特殊情况的渲染（不支持材质编辑器）
- 我们的方案：把整个渲染管线挪到桌面OpenGL的代码上
 - 新的RHI(渲染硬件接口) OpenGL分支使用了EGL支持OpenGL 3.2 和OpenGL ES 3.1

安卓调试



调试面板(D3D)

The screenshot displays the NVIDIA D3D debug panel with several key components:

- Scrubber:** A timeline at the top showing frame numbers from 22000 to 33000. It includes sections for Events, Action, Dependencies, and Perf Markers.
- Current Target:** A 3D view of a game scene with a character, overlaid with a red line indicating the current target.
- Events:** A list of GPU events, such as `IDirect3DDevice9::SetRenderState(D3DRENDERSTATE_FO...`, with a search filter set to `DDDD`.
- API Inspector:** A detailed view of the selected event, showing the `Call Description` and `Arguments`.
- Parallel Shader:** A visual representation of the shader's execution flow, showing a triangular pattern of green dots.
- Resources:** A window showing the `Texture 114` resource, including its dimensions (128x128) and memory usage.
- Locals:** A window for inspecting local variables.
- Output:** A log window showing messages like `Shader created` with unique IDs.

调试面板 (PC GL)

The screenshot displays the NVIDIA DebugView application interface, which is used for debugging graphics drivers and applications. The interface is divided into several main sections:

- Timeline (Top):** Shows a sequence of events and actions over time, with a scrubber at the top. Key events include 'Shadowed Lights', 'Unshadowed Lights', 'Decals/Translucent', 'PostProcessEffects', 'Transparency', 'DNG Forest', 'PostProcessEffects', 'RenderScaleBox', and 'BeginDisplay'. A 'Perf Markers' section shows performance markers for 'DominantDirectionalLight_u' and 'Lights'.
- Resources (Right):** Displays a 3D scene with a character and a red laser beam. A 'Current Target' dropdown is set to 'Color_2.P'.
- Events (Bottom Left):** A list of events with columns for 'Event', 'Description', and 'Context'. The selected event is '8514 P' with the description 'gDrawRangeElements(GLenum mode = GL_TRIANGLES, GLuint start = 0, GLuint end = 4, GLsizei count = ...)'.
- API Inspector (Middle Left):** Shows the 'OpenGL.Drv.h' API call description for 'gDrawRangeElements'. It includes details for 'Vtx Spec', 'Transform', 'VS', 'TCS', 'TES', 'GS', 'NFB', 'Raster', 'FS', 'Pix Ops', 'FB', 'CS', 'Textures', 'Images', 'Buffers', 'Program', 'Pixels', and 'Misc'.
- Program Shader (Middle Right):** Displays the 'Attached Shader State' for the selected event, showing 'Name: 4791 Shader.P', 'Delete Status: GL_TRUE', and 'Compile Status: GL_TRUE'.
- Resources (Bottom Right):** Shows a 'Viewer Focus: Hp Chain' with a 3D visualization of a green, glowing, V-shaped object. It includes 'Resource Info' and 'Available Revisions (1)'.
- Locals (Bottom Left):** A table for local variables with columns for 'Name', 'Value', and 'Type'.
- Output (Bottom Right):** A text area showing system messages and error logs, including a warning about 'The target system's TDR delay is set to 2 seconds'.

调试面板 (SHIELD)

The screenshot displays the NVIDIA SHIELD debugging interface, which is used for analyzing GPU performance and rendering issues. The interface is divided into several key sections:

- Scrubber View:** Shows a timeline of GPU events and performance markers. The top bar indicates the current frame number (9111) and a zoomed-in view of the GPU clock (7900-9200). Below this, various rendering stages are visualized as colored bars, including UnshadedVerts, Light, Decals, PostProcess, Distortion, Transparency, and RenderScale. Performance markers for different components like SphericalHar, Light, and Dist are also shown.
- Events View:** A list of GPU events with columns for Event ID, Description, Context, CPU us, and GPU us. The list shows a sequence of rendering commands such as `glDrawBuffers`, `glViewport`, `glInvalidateTexSubImage`, and `glBegin`.
- API Inspector View:** Provides a detailed view of the current API call. The left pane shows the function signature (e.g., `void glDrawRangeElements`). The right pane shows the call description, including shader state, samplers, and program interfaces.
- Resources View:** Displays a 3D visualization of the GPU resources used by the current draw call. In this case, it shows a large, glowing cyan heart shape composed of many small points, representing the data being rendered.
- Current Target View:** Shows the rendered scene, which is a 3D environment with a character and various structures.



小结

- 现在，我们有了一个功能完善，可以跑起来的游戏版本了
- 游戏帧数还有高有低
- 下一步就是优化游戏性能，这样玩起来才更爽！



初始性能总结

- 在渲染线程上既有CPU瓶颈又有GPU瓶颈
- 相比于PC，安卓上有更多的卡顿
- 场景1: 主菜单
 - 15-30FPS, 1080p下有GPU瓶颈
- 场景2: 第一关
 - 26-28FPS, 1080p下CPU/GPU 都有瓶颈
 - Null GPU 可以跑到40FPS
- 主线程在TX1上的消耗在10-12ms



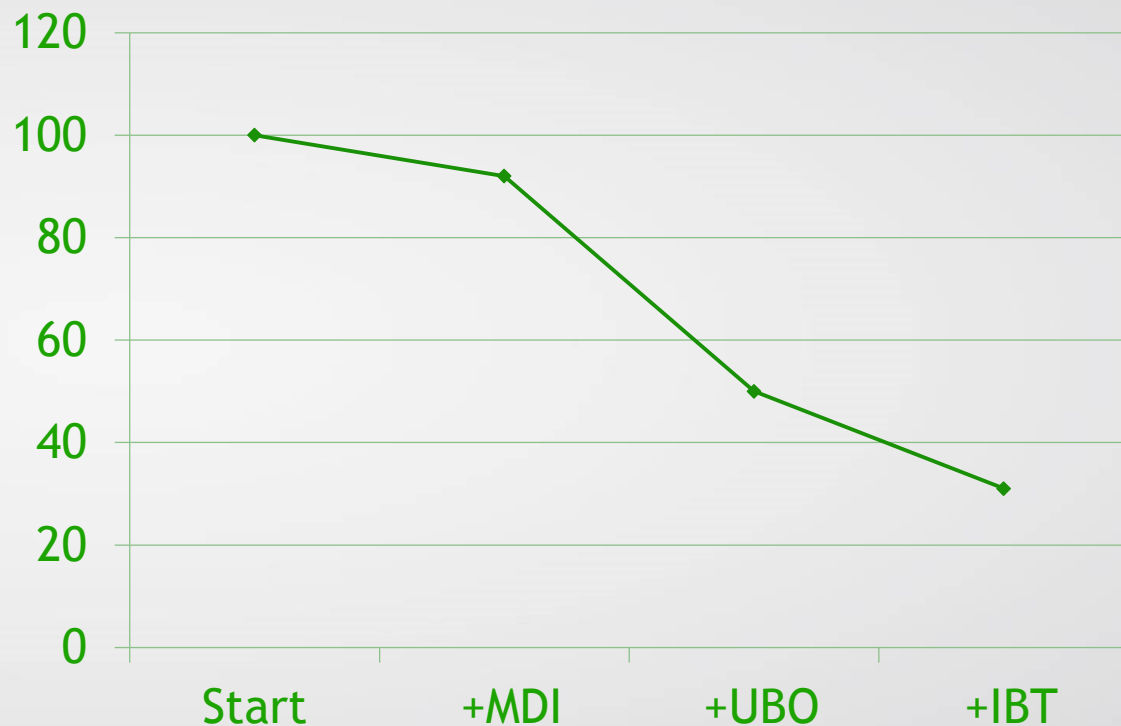
UE3 OpenGL集成代码分析

- OpenGL 3.2
- 好处
 - 和D3D9的API一一对应
 - 简单易用
 - API 自己的特性
- 坏处
 - 性能不够好
- 有很多方法可以提高OpenGL API的效率
 - 参见 GDC 2014的演讲“Approaching Zero Driver Overhead”



OpenGL API 优化

- 第一关测试场景
 - 调试器显示10K+事件
 - 1636 次绘图调用(Draw Call)
- 使用MultiDrawIndirect
 - 1494次绘图调用(Draw Call)
 - 减少了8%
- 正在进行的工作...
- 一个大的被顶点索引的UBO
 - 809 绘图调用(Draw Call)
 - 减少了50%
 - 减少了uniform常数的绑定次数
- Indexed bindless textures
 - 502 绘图调用 (Draw Call)
 - 减少了69%



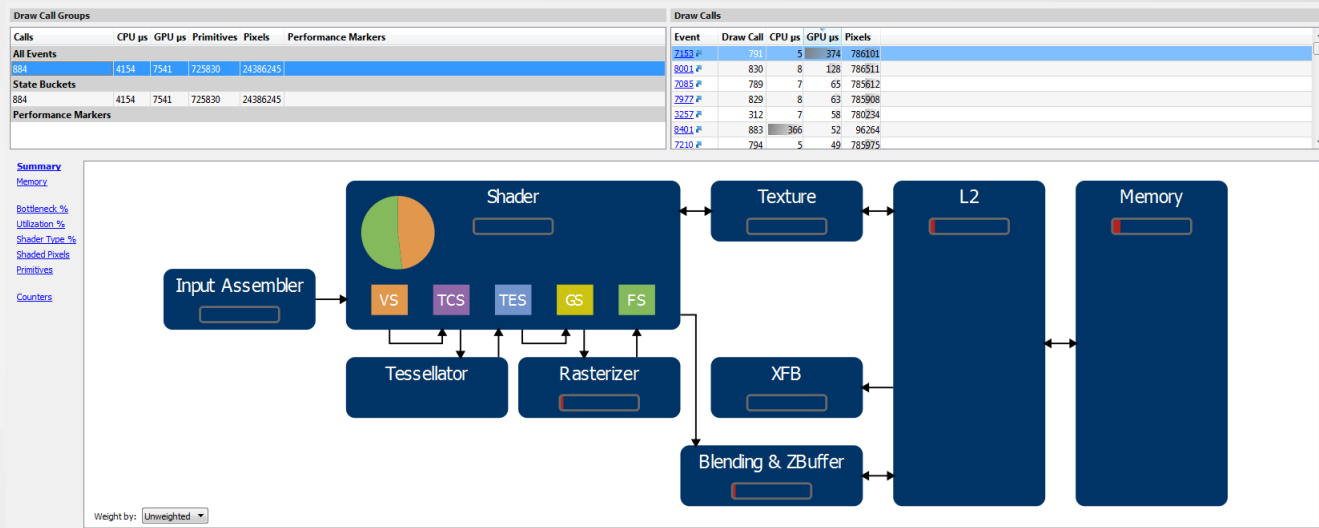
GPU 性能优化

- 正在进行的GPU瓶颈分析

- 可能用到的技术

- Z-prepass
- 后处理质量
- 反走样 (AA) 的选择
- 分辨率大小
- Shader优化

- 技术的使用根据实际需求确定



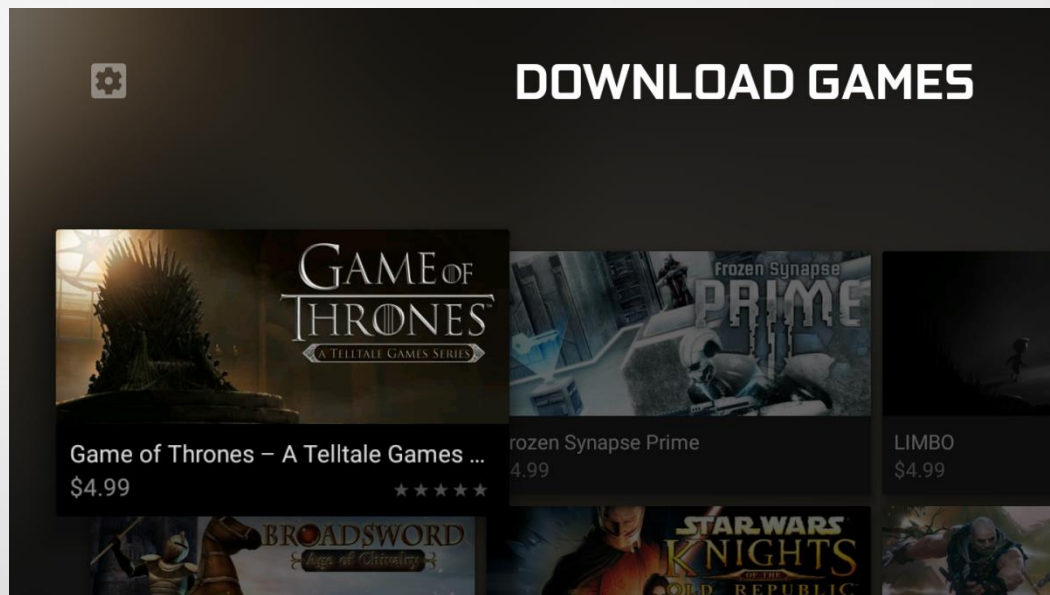
其它的工作

- Debug context
- OpenGL ES上对BGRA的支持
- OpenGL对D3D里PF_V8U8 bump map的支持
- ARM NEON浮点运算加速的支持
- Array cookie 大小的不同
- 内存对齐的问题



联网功能

- 修改代码适应Google Play
- 多人对战
- 云存储，成就系统
- DLC/附加内容
- Google Play API的集成



《无主之地：前传》视频



QA

