



# DEVELOPER TOOLS FOR MOBILE PLATFORMS

Jonas Yang, Manager of Mobile Graphics Tools

Introducing NVIDIA® Tegra Graphics Debugger 2.0

## AGENDA

Profiling UE4 SumTemple Demo

Introducing Tegra System Profiler



# NVIDIA DEVELOPER TOOLS

BUILD. DEBUG. PROFILE.

Microsoft  
DirectX

OpenGL

nVIDIA  
CUDA

OpenGL|ES



IDE INTEGRATION

Visual Studio  
eclipse

STANDALONE TOOLS



HARDWARE SUPPORT  
CPU AND GPU DEBUGGING & PROFILING



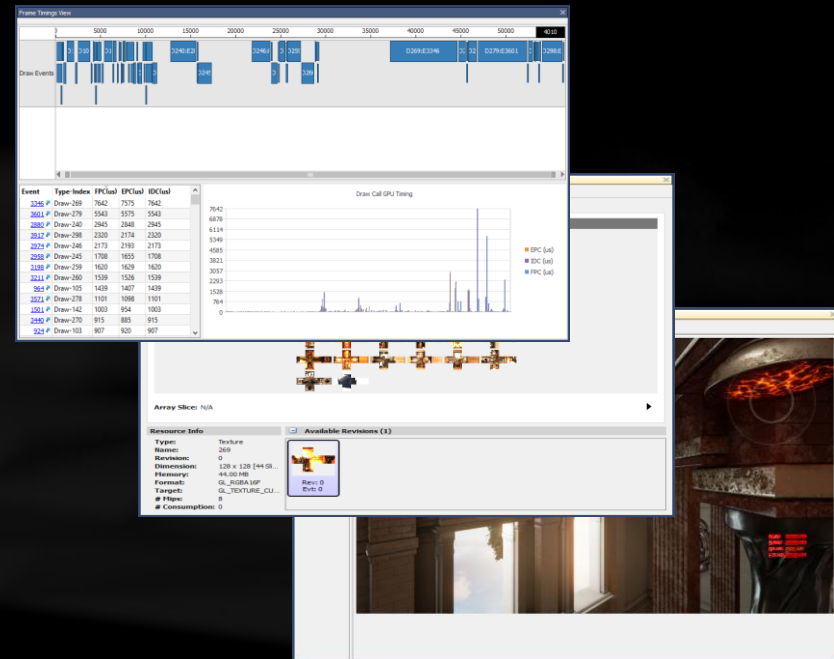
nVIDIA  
GAMEWORKS™



# NVIDIA® TEGRA GRAPHICS DEBUGGER

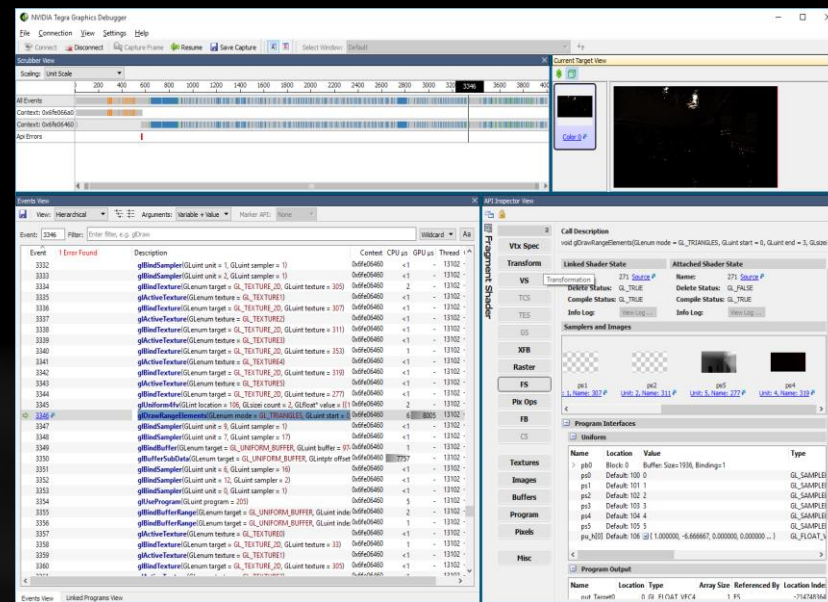
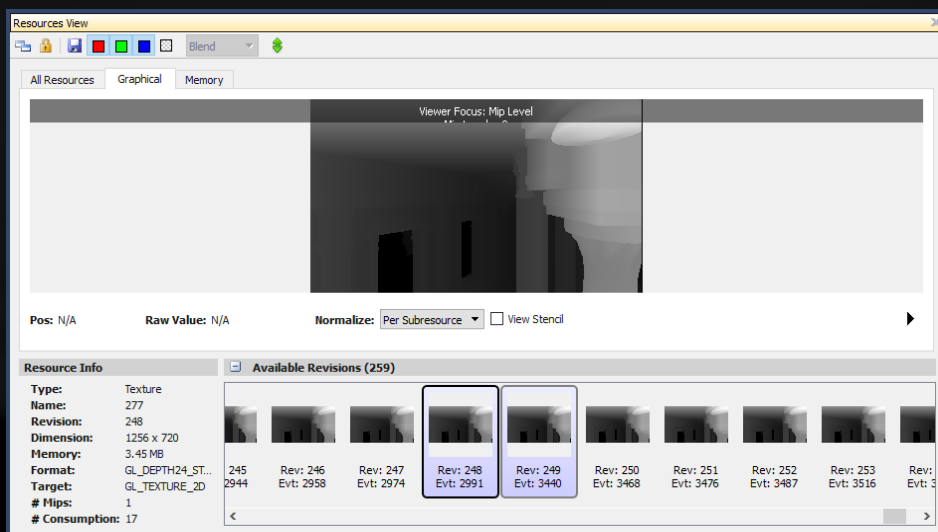
Accelerating Visual Computing Development on *Mobile Platforms*

- Supports cutting edge OpenGL
- Debug and profile graphics workloads
- Live capture of a single rendering frame
- Automatic GPU bottleneck analysis
- Advanced timings for draw calls
- Standalone tool on Windows/Mac/Linux



# NEW IN VERSION 2.0

- Supports OpenGL ES 3.1 and OpenGL 4.5
- Support Android 5.0 and Android TV
- Support arm 32bit/64bit Android
- Serialize captured frame
- Target HUD



# PROFILING UE4 SUNTEMPLE DEMO

Connect to Application

Target: SHIELD Tablet - 1740CA109100000009060340

Launch APK Attach

Search for an APK...

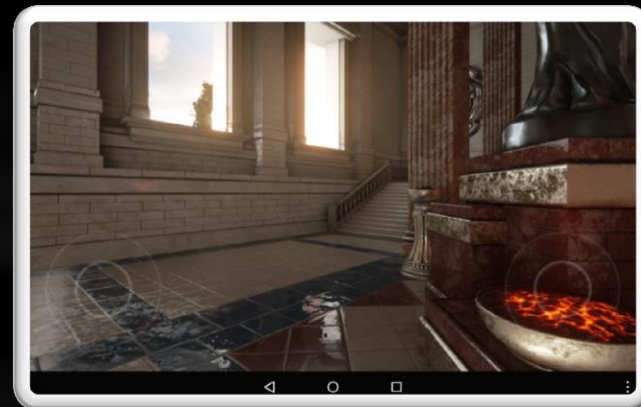
Package Name	Package Location
com.nvidia.ParticleUpsampling	/data/app/com.nvidia.ParticleUpsampling-1/base.apk
com.nvidia.UnityWindmill2ES3001	/data/app/com.nvidia.UnityWindmill2ES3001-1/base.apk
com.nvidia.demo	/data/app/com.nvidia.demo-1/base.apk
com.nvidia.main_2015_07_27_23_26_33	/data/app/com.nvidia.main_2015_07_27_23_26_33-2/base.apk
com.nvidia.main_2015_07_28_15_17_24	/data/app/com.nvidia.main_2015_07_28_15_17_24-1/base.apk
com.nvidia.tegraprofiler.tools	/data/app/com.nvidia.tegraprofiler.tools-1/base.apk
com.ru.golf	/data/app/com.ru.golf-1/base.apk
com.ue4.SunTemple47	/data/app/com.ue4.SunTemple47-1/base.apk
com.vectorunit.purple.googleplay	/data/app/com.vectorunit.purple.googleplay-1/base.apk
com.walnut.striveNvidia	/data/app/com.walnut.striveNvidia-1/base.apk
net.kishonti.compubench.gl.v10500.corporate	/data/app/net.kishonti.compubench.gl.v10500.corporate-1/base.apk

Status:

Device Product Model: SHIELD Tablet  
Making sure adb has root permissions  
Waiting for the device to become available.  
Mounting device for read/write.  
Detected current device is running Android L  
Setting SELinux policy to permissive.  
SELinux policy successfully changed to permissive.  
Located 32-bit driver libraries in /vendor/lib  
Host supports 32-bit graphics debugging  
Detected NVIDIA tegra device drivers  
Graphics debugger version is correct.  
Searching for packages installed on the current device.  
Scanning for attachable processes on SHIELD Tablet - 1740CA109100000009060340...  
Scan for devices complete.  
Process scan complete.

ADB: F:\NVPACK\android-sdk-windows\platform-tools\adb.exe Browse... Cancel Launch

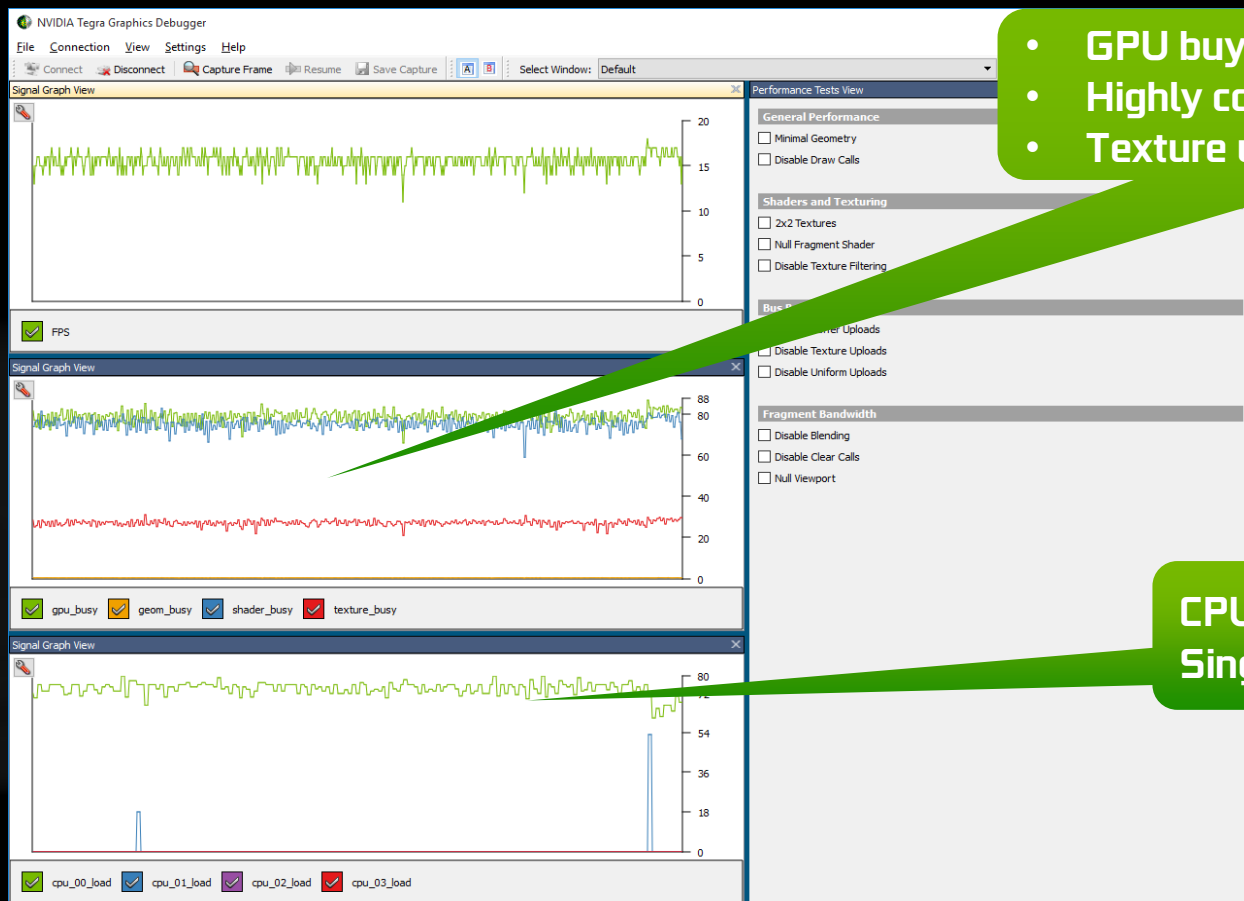
APK Name



Launch!



# PROFILING UE4 DEMO

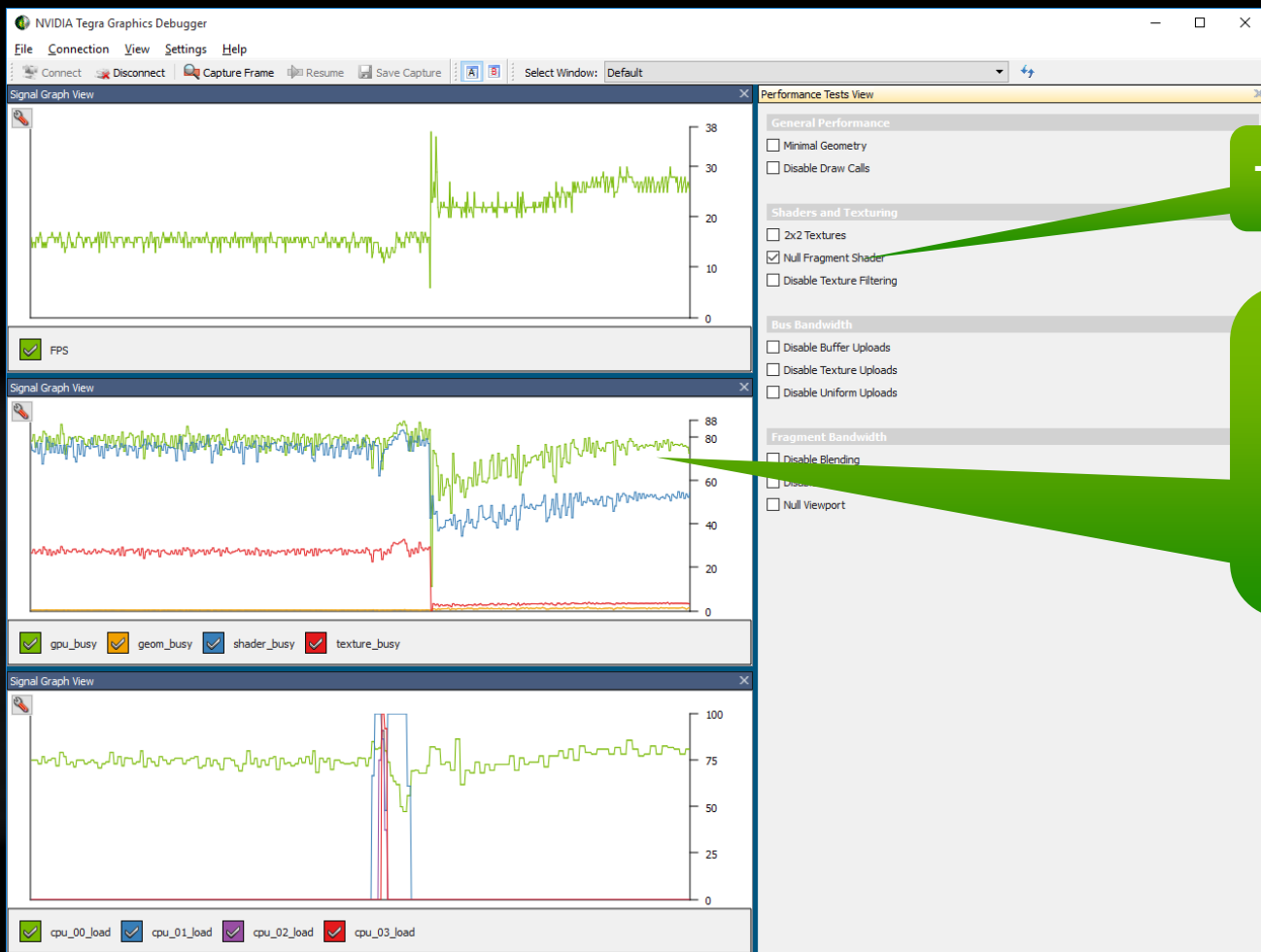


- GPU busy is 80%
- Highly correlated to shader unit busy
- Texture unit is not so busy 30%

CPU is 80% loaded  
Single core is utilized



# QUICK EXPERIMENTS



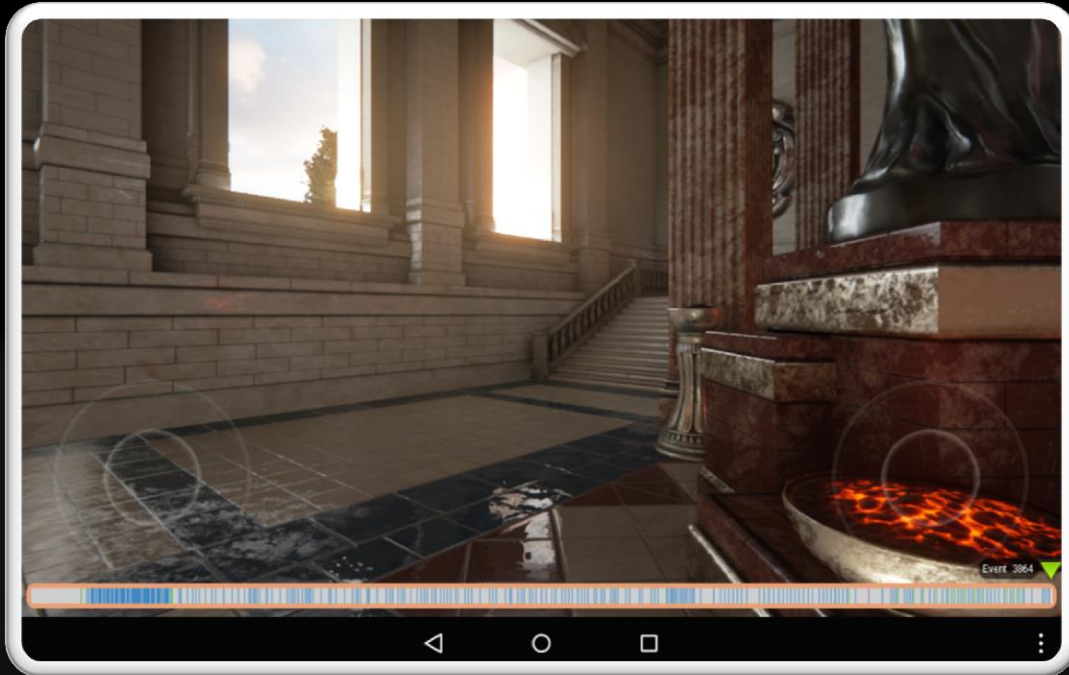
Toggle Null Fragment Shader

What?  
GPU is still busy?  
Shader unit is still busy?  
Answer: Compute shader is used!!!





# CAPTURE A FRAME



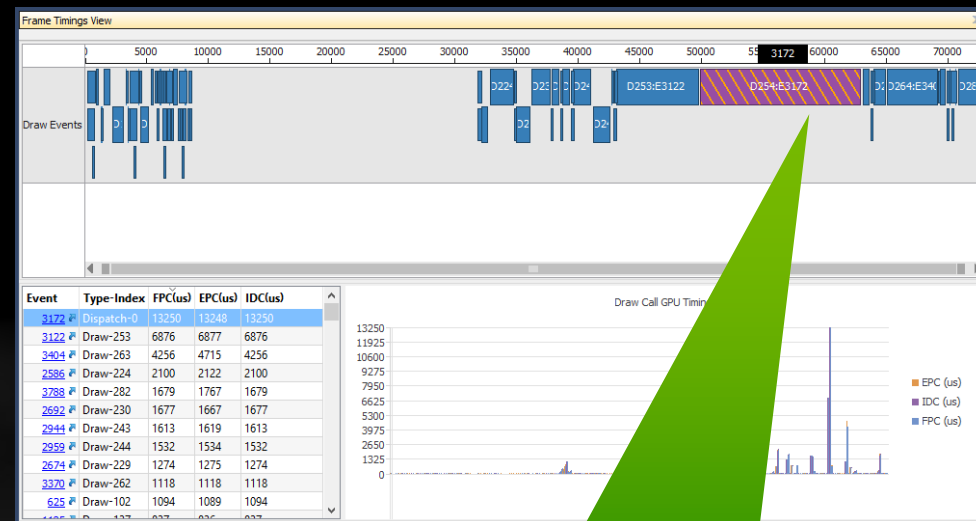
The screenshot displays the NVIDIA Nsight Graphics Debugger interface. The main window shows the Resources View, which lists various resources such as Drawables, Textures, Renderbuffers, Buffers, and Thumbnails. The Events View shows a list of events, including GL calls like glUseProgram, glBindTexture, and glDrawRangeElements. The Resources View also shows details for the selected resource, including its type, name, dimensions, and memory consumption. The interface is divided into several panels, including the Resources View, Events View, and a detailed view of the selected resource.



# INSPECT EXPENSIVE APIS

The screenshot shows the NVIDIA Tegra Graphics Debugger interface. The top window is the Scrubber View, showing a timeline of events. The middle window is the Events View, displaying a list of events with columns for Event, Description, Context, CPU  $\mu$ s, GPU  $\mu$ s, Thread, and Queue. The bottom window is the API Inspector View, showing the details for the selected event (3172), including the call description, Vtx Spec, and Program Interfaces.

Event	Description	Context	CPU $\mu$ s	GPU $\mu$ s	Thread	Queue
3172 #	glDispatchCompute(GLuint num_groups_x = 79, GLuint num_groups_y = 5, GLuint num_groups_z = 1)	0x57b06490	199	13248	6233	-
3122 #	glDrawRangeElements(GLenum mode = GL_TRIANGLES, GLuint start = 0, GLuint end = 1000000, GLsizei count = 1000000, GLenum type = GL_UNSIGNED_SHORT, GLsizei offset = 0)	0x57b06490	8	8877	6233	-
3404 #	glDrawRangeElements(GLenum mode = GL_TRIANGLES, GLuint start = 0, GLuint end = 1000000, GLsizei count = 1000000, GLenum type = GL_UNSIGNED_SHORT, GLsizei offset = 0)	0x57b06490	9	4715	6233	-
2586 #	glDrawRangeElements(GLenum mode = GL_TRIANGLES, GLuint start = 0, GLuint end = 1000000, GLsizei count = 1000000, GLenum type = GL_UNSIGNED_SHORT, GLsizei offset = 0)	0x57b06490	8	2122	6233	-
3788 #	glDrawRangeElements(GLenum mode = GL_TRIANGLES, GLuint start = 0, GLuint end = 1000000, GLsizei count = 1000000, GLenum type = GL_UNSIGNED_SHORT, GLsizei offset = 0)	0x57b06490	7	1767	6233	-
2692 #	glDrawRangeElements(GLenum mode = GL_TRIANGLES, GLuint start = 0, GLuint end = 1000000, GLsizei count = 1000000, GLenum type = GL_UNSIGNED_SHORT, GLsizei offset = 0)	0x57b06490	7	1667	6233	-
2944 #	glDrawArraysInstanced(GLenum mode = GL_TRIANGLE_STRIP, GLint first = 0, GLsizei count = 1000000, GLuint instance_start = 0, GLuint instance_count = 1000000)	0x57b06490	11	1619	6233	-
2659 #	glDrawRangeElements(GLenum mode = GL_TRIANGLES, GLuint start = 0, GLuint end = 1000000, GLsizei count = 1000000, GLenum type = GL_UNSIGNED_SHORT, GLsizei offset = 0)	0x57b06490	7	1633	6233	-
3370 #	glDrawRangeElements(GLenum mode = GL_TRIANGLES, GLuint start = 0, GLuint end = 1000000, GLsizei count = 1000000, GLenum type = GL_UNSIGNED_SHORT, GLsizei offset = 0)	0x57b06490	8	1633	6233	-
625 #	glDrawRangeElements(GLenum mode = GL_TRIANGLES, GLuint start = 0, GLuint end = 1000000, GLsizei count = 1000000, GLenum type = GL_UNSIGNED_SHORT, GLsizei offset = 0)	0x57b06490	23	1000	6233	-
1183 #	glDrawRangeElements(GLenum mode = GL_TRIANGLES, GLuint start = 0, GLuint end = 1000000, GLsizei count = 1000000, GLenum type = GL_UNSIGNED_SHORT, GLsizei offset = 0)	0x57b06490	13	836	6233	-
2227 #	glDrawArraysInstanced(GLenum mode = GL_TRIANGLES_STRIP, GLint first = 0, GLsizei count = 1000000, GLuint instance_start = 0, GLuint instance_count = 1000000)	0x57b06490	13	775	6233	-
2813 #	glDrawArraysInstanced(GLenum mode = GL_TRIANGLE_STRIP, GLint first = 0, GLsizei count = 1000000, GLuint instance_start = 0, GLuint instance_count = 1000000)	0x57b06490	11	778	6233	-
3225 #	glDrawRangeElements(GLenum mode = GL_TRIANGLES, GLuint start = 0, GLuint end = 1000000, GLsizei count = 1000000, GLenum type = GL_UNSIGNED_SHORT, GLsizei offset = 0)	0x57b06490	10	744	6233	-
610 #	glDrawRangeElements(GLenum mode = GL_TRIANGLES, GLuint start = 0, GLuint end = 1000000, GLsizei count = 1000000, GLenum type = GL_UNSIGNED_SHORT, GLsizei offset = 0)	0x57b06490	16	708	6233	-
2534 #	glDrawArraysInstanced(GLenum mode = GL_TRIANGLE_STRIP, GLint first = 0, GLsizei count = 1000000, GLuint instance_start = 0, GLuint instance_count = 1000000)	0x57b06490	17	655	6233	-
3451 #	glDrawRangeElements(GLenum mode = GL_TRIANGLES, GLuint start = 0, GLuint end = 1000000, GLsizei count = 1000000, GLenum type = GL_UNSIGNED_SHORT, GLsizei offset = 0)	0x57b06490	12	594	6233	-
1726 #	glDrawRangeElements(GLenum mode = GL_TRIANGLES, GLuint start = 0, GLuint end = 1000000, GLsizei count = 1000000, GLenum type = GL_UNSIGNED_SHORT, GLsizei offset = 0)	0x57b06490	10	477	6233	-
566 #	glDrawRangeElements(GLenum mode = GL_TRIANGLES, GLuint start = 0, GLuint end = 1000000, GLsizei count = 1000000, GLenum type = GL_UNSIGNED_SHORT, GLsizei offset = 0)	0x57b06490	16	392	6233	-
1213 #	glDrawRangeElements(GLenum mode = GL_TRIANGLES, GLuint start = 0, GLuint end = 1000000, GLsizei count = 1000000, GLenum type = GL_UNSIGNED_SHORT, GLsizei offset = 0)	0x57b06490	10	355	6233	-
2265 #	glDrawRangeElements(GLenum mode = GL_TRIANGLES, GLuint start = 0, GLuint end = 1000000, GLsizei count = 1000000, GLenum type = GL_UNSIGNED_SHORT, GLsizei offset = 0)	0x57b06490	9	290	6233	-
521 #	glDrawRangeElements(GLenum mode = GL_TRIANGLES, GLuint start = 0, GLuint end = 1000000, GLsizei count = 1000000, GLenum type = GL_UNSIGNED_SHORT, GLsizei offset = 0)	0x57b06490	17	271	6233	-
1311 #	glDrawRangeElements(GLenum mode = GL_TRIANGLES, GLuint start = 0, GLuint end = 1000000, GLsizei count = 1000000, GLenum type = GL_UNSIGNED_SHORT, GLsizei offset = 0)	0x57b06490	9	270	6233	-
2983 #	glDrawRangeElements(GLenum mode = GL_TRIANGLES, GLuint start = 0, GLuint end = 1000000, GLsizei count = 1000000, GLenum type = GL_UNSIGNED_SHORT, GLsizei offset = 0)	0x57b06490	7	230	6233	-
628 #	glDrawRangeElements(GLenum mode = GL_TRIANGLES, GLuint start = 0, GLuint end = 1000000, GLsizei count = 1000000, GLenum type = GL_UNSIGNED_SHORT, GLsizei offset = 0)	0x57b06490	22	202	6233	-
3229 #	glDrawRangeElements(GLenum mode = GL_TRIANGLES, GLuint start = 0, GLuint end = 1000000, GLsizei count = 1000000, GLenum type = GL_UNSIGNED_SHORT, GLsizei offset = 0)	0x57b06490	7	196	6233	-
1700 #	glDrawRangeElements(GLenum mode = GL_TRIANGLES, GLuint start = 0, GLuint end = 1000000, GLsizei count = 1000000, GLenum type = GL_UNSIGNED_SHORT, GLsizei offset = 0)	0x57b06490	9	194	6233	-
1136 #	glDrawRangeElements(GLenum mode = GL_TRIANGLES, GLuint start = 0, GLuint end = 1000000, GLsizei count = 1000000, GLenum type = GL_UNSIGNED_SHORT, GLsizei offset = 0)	0x57b06490	9	191	6233	-
637 #	glDrawRangeElements(GLenum mode = GL_TRIANGLES, GLuint start = 0, GLuint end = 1000000, GLsizei count = 1000000, GLenum type = GL_UNSIGNED_SHORT, GLsizei offset = 0)	0x57b06490	28	177	6233	-



No parallel GPU workload at all!

Most expensive API is compute shading!



# INSPECT EXPENSIVE APIS



NVIDIA Tegra Graphics Debugger

Events View

Event	Description	Context	CPU $\mu$ s	GPU $\mu$ s	Thread	Queue
3107	glBindSampler(GLuint unit = 0, GLuint sampler = 1)	0x7b06490	1	-	6233	-
3108	glBindSampler(GLuint unit = 1, GLuint sampler = 1)	0x7b06490	<1	-	6233	-
3109	glBindSampler(GLuint unit = 2, GLuint sampler = 1)	0x7b06490	<1	-	6233	-
3110	glBindTexture(GLenum target = GL_TEXTURE_2D, GLuint texture = 306)	0x7b06490	2	-	6233	-
3111	glActiveTexture(GLenum texture = GL_TEXTURE1)	0x7b06490	<1	-	6233	-
3112	glBindTexture(GLenum target = GL_TEXTURE_2D, GLuint texture = 307)	0x7b06490	2	-	6233	-
3113	glActiveTexture(GLenum texture = GL_TEXTURE2)	0x7b06490	<1	-	6233	-
3114	glBindTexture(GLenum target = GL_TEXTURE_2D, GLuint texture = 309)	0x7b06490	1	-	6233	-
3115	glActiveTexture(GLenum texture = GL_TEXTURE3)	0x7b06490	<1	-	6233	-
3116	glBindTexture(GLenum target = GL_TEXTURE_2D, GLuint texture = 337)	0x7b06490	1	-	6233	-
3117	glActiveTexture(GLenum texture = GL_TEXTURE4)	0x7b06490	<1	-	6233	-
3118	glBindTexture(GLenum target = GL_TEXTURE_2D, GLuint texture = 313)	0x7b06490	1	-	6233	-
3119	glActiveTexture(GLenum texture = GL_TEXTURE5)	0x7b06490	<1	-	6233	-
3120	glBindTexture(GLenum target = GL_TEXTURE_2D, GLuint texture = 270)	0x7b06490	1	-	6233	-
3121	glInvalidate(GLint location = 106, GLsizei count = 2, GLfloat* values = {})	0x7b06490	1	-	6233	-
3122	glDrawRangeElements(GLenum mode = GL_TRIANGLES, GLuint start = 0)	0x7b06490	8	8877	6233	-
3123	glBindSampler(GLuint unit = 9, GLuint sampler = 1)	0x7b06490	2	-	6233	-
3124	glBindSampler(GLuint unit = 7, GLuint sampler = 17)	0x7b06490	1	-	6233	-
3125	eglGetCurrentContext() = 0x7b06490	0x7b06490	15	-	6233	-
3126	glBindBuffer(GLenum target = GL_UNIFORM_BUFFER, GLuint buffer = 38)	0x7b06490	3	-	6233	-
3127	glBufferSubData(GLenum target = GL_UNIFORM_BUFFER, GLintptr offset = 0)	0x7b06490	30	-	6233	-
3128	glBindSampler(GLuint unit = 6, GLuint sampler = 10)	0x7b06490	4	-	6233	-
3129	glBindSampler(GLuint unit = 12, GLuint sampler = 2)	0x7b06490	3	-	6233	-
3130	glBindSampler(GLuint unit = 0, GLuint sampler = 1)	0x7b06490	2	-	6233	-
3131	eglGetCurrentContext() = 0x7b06490	0x7b06490	4	-	6233	-
3132	glUseProgram(GLuint program = 255)	0x7b06490	9	-	6233	-
3133	glBindBufferRange(GLenum target = GL_UNIFORM_BUFFER, GLuint index = 0)	0x7b06490	4	-	6233	-
3134	glBindBufferRange(GLenum target = GL_UNIFORM_BUFFER, GLuint index = 0)	0x7b06490	3	-	6233	-
3135	glActiveTexture(GLenum texture = GL_TEXTURE0)	0x7b06490	1	-	6233	-
3136	glBindTexture(GLenum target = GL_TEXTURE_2D, GLuint texture = 340)	0x7b06490	5	-	6233	-

API Inspector View

Call Description  
void glDrawRangeElements(GLenum mode = GL\_TRIANGLES, GLuint start = 0, GLuint end = 3, GLsizei

Transform

VS  
TCS  
TES  
GS  
XFB  
Raster  
FOS  
Pix Ops  
FB  
CS

Program Interfaces

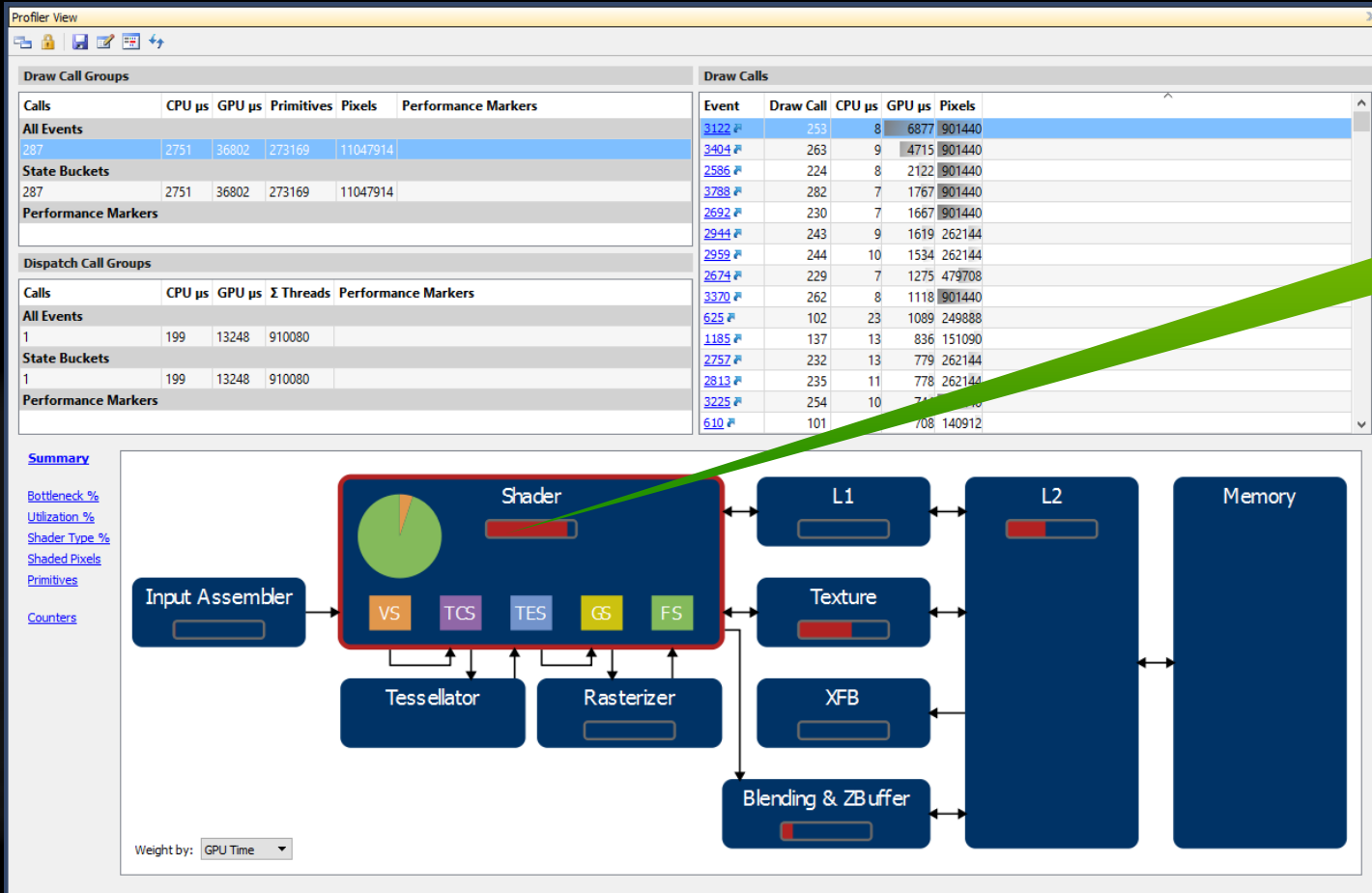
Name	Location	Value	Type
pb0	Block: 0	Buffer: Size=1936, Bindings=1	GL_SAMPLER
ps0	Default: 100 0		GL_SAMPLER
ps1	Default: 101 1		GL_SAMPLER
ps2	Default: 102 2		GL_SAMPLER
ps3	Default: 103 3		GL_SAMPLER
ps4	Default: 104 4		GL_SAMPLER
ps5	Default: 105 5		GL_SAMPLER
psu[0]	Default: 106 6	{ 1.000000, -6.666667, 0.000000, 0.000000 ... }	GL_FLOAT_V

Program Output

Name	Location	Type	Array Size	Referenced By	Location Index
out_Turnoff	0	GL_FLOAT_VEC4	1 PS		-714748164



# RUN PROFILER



Almost all shader bottleneck



# SERIALIZE A FRAME

The screenshot displays the NVIDIA Tegra Graphics Debugger interface. The top section shows a timeline of events with a red highlight on event 3122. The Events View table is as follows:

Event	Description	Context	CPU us	GPU us	Thread	Queue
3122	glDrawRangeElements(Glenum mode = GL_TRIANGLES, GLuint start = 0, GLuint count = 17, GLuint instance = 0, GLuint instance_max = 0)	0x57b06490	8	6877	6233	-
3123	glBindSampler(GLuint unit = 9, GLuint sampler = 1)	0x57b06490	2	-	6233	-
3124	glBindSampler(GLuint unit = 7, GLuint sampler = 17)	0x57b06490	1	-	6233	-
3125	eglGetCurrentContext() = 0x57b06490	0x57b06490	15	-	6233	-
3126	glBindBuffer(GLenum target = GL_UNIFORM_BUFFER, GLuint buffer = 39, GLintptr offset = 0)	0x57b06490	3	-	6233	-
3127	glBufferSubData(GLenum target = GL_UNIFORM_BUFFER, GLintptr offset = 0, GLvoid* data, GLsizeiptr size)	0x57b06490	30	-	6233	-
3128	glBindSampler(GLuint unit = 6, GLuint sampler = 16)	0x57b06490	4	-	6233	-
3129	glBindSampler(GLuint unit = 12, GLuint sampler = 2)	0x57b06490	3	-	6233	-
3130	glBindSampler(GLuint unit = 0, GLuint sampler = 1)	0x57b06490	2	-	6233	-
3131	eglGetCurrentContext() = 0x57b06490	0x57b06490	4	-	6233	-
3132	glUseProgram(GLuint program = 265)	0x57b06490	9	-	6233	-
3133	glBindBufferRange(GLenum target = GL_UNIFORM_BUFFER, GLuint index = 0, GLvoid* data, GLsizeiptr size)	0x57b06490	4	-	6233	-
3134	glBindBufferRange(GLenum target = GL_UNIFORM_BUFFER, GLuint index = 1, GLvoid* data, GLsizeiptr size)	0x57b06490	3	-	6233	-
3135	glActiveTexture(GLenum texture = GL_TEXTURE0)	0x57b06490	1	-	6233	-
3136	glBindTexture(GLenum target = GL_TEXTURE_2D, GLuint texture = 34)	0x57b06490	5	-	6233	-
3137	glActiveTexture(GLenum texture = GL_TEXTURE1)	0x57b06490	1	-	6233	-
3138	glBindTexture(GLenum target = GL_TEXTURE_2D, GLuint texture = 306)	0x57b06490	3	-	6233	-
3139	glActiveTexture(GLenum texture = GL_TEXTURE2)	0x57b06490	1	-	6233	-
3140	glBindTexture(GLenum target = GL_TEXTURE_2D, GLuint texture = 307)	0x57b06490	2	-	6233	-
3141	glActiveTexture(GLenum texture = GL_TEXTURE3)	0x57b06490	1	-	6233	-
3142	glBindTexture(GLenum target = GL_TEXTURE_2D, GLuint texture = 308)	0x57b06490	4	-	6233	-
3143	glActiveTexture(GLenum texture = GL_TEXTURE4)	0x57b06490	1	-	6233	-
3144	glBindTexture(GLenum target = GL_TEXTURE_2D, GLuint texture = 309)	0x57b06490	3	-	6233	-
3145	glActiveTexture(GLenum texture = GL_TEXTURE5)	0x57b06490	1	-	6233	-
3146	glBindTexture(GLenum target = GL_TEXTURE_2D, GLuint texture = 311)	0x57b06490	4	-	6233	-
3147	glActiveTexture(GLenum texture = GL_TEXTURE6)	0x57b06490	1	-	6233	-
3148	glBindTexture(GLenum target = GL_TEXTURE_2D, GLuint texture = 277)	0x57b06490	4	-	6233	-
3149	glActiveTexture(GLenum texture = GL_TEXTURE7)	0x57b06490	1	-	6233	-
3150	glBindTexture(GLenum target = GL_TEXTURE_CUBE_MAP_ARRAY_EXT, GLuint texture = 0)	0x57b06490	4	-	6233	-
3151	glActiveTexture(GLenum texture = GL_TEXTURE8)	0x57b06490	1	-	6233	-

The API Inspector View for event 3122 shows the following details:

- Call Description: void glDrawRangeElements(GLenum mode = GL\_TRIANGLES, GLuint start = 0, GLuint end = 3, GLsizei ...)
- Transform: VS (Vertex Shader)
- Program Interfaces: Uniform
- Textures: pb0 (Block: 0, Buffer: Size=1936, Bindings=1), ps0 (Default: 100 0, GL\_SAMPLER), ps1 (Default: 101 1, GL\_SAMPLER), ps2 (Default: 102 2, GL\_SAMPLER), ps3 (Default: 103 3, GL\_SAMPLER), ps4 (Default: 104 4, GL\_SAMPLER), ps5 (Default: 105 5, GL\_SAMPLER), pu\_n[0] (Default: 106, [ 1.000000, -6.666667, 0.000000, 0.000000 ... ], GL\_FLOAT\_V)

Almost all shader bottleneck



# OPEN THE SOLUTION FILE

Name	Date modified	Type	Size
android-kk-egl-t124-a32	7/28/2015 12:01 PM	File folder	
android-L-egl-t132-a64	7/28/2015 12:01 PM	File folder	
assets	7/28/2015 12:02 PM	File folder	
res	7/28/2015 12:17 PM	File folder	
Tegra-Android	7/28/2015 12:24 PM	File folder	
android_native_app_glue.c	7/28/2015 11:29 AM	C Source file	17 KB
android_native_app_glue.h	7/28/2015 11:29 AM	C++ Header file	12 KB
AndroidManifest.xml	7/28/2015 11:29 AM	XML Document	2 KB
egl.h	7/28/2015 11:29 AM	C++ Header file	16 KB
eglex.h	7/28/2015 11:29 AM	C++ Header file	41 KB
Frame0Part00.cpp	7/28/2015 11:29 AM	C++ Source file	397 KB
FrameReset00.cpp	7/28/2015 11:29 AM	C++ Source file	115 KB
FrameSetup00.cpp	7/28/2015 11:29 AM	C++ Source file	300 KB
FrameSetup01.cpp	7/28/2015 11:29 AM	C++ Source file	570 KB
FrameSetup02.cpp	7/28/2015 11:29 AM	C++ Source file	384 KB
gl2ext.h	7/28/2015 11:29 AM	C++ Header file	168 KB
gl31.h	7/28/2015 11:29 AM	C++ Header file	
Helpers.cpp	7/28/2015 11:29 AM	C++ Source file	5 KB
Helpers.h	7/28/2015 11:29 AM	C++ Header file	4 KB
Main.cpp	7/28/2015 11:29 AM	C++ Source file	5 KB
main_2015_07_27_23_26_33_vs2010.sln	7/28/2015 11:29 AM	Microsoft Visual S...	2 KB
main_2015_07_27_23_26_33_vs2010.vcx...	7/28/2015 11:29 AM	VC++ Project	8 KB
main_2015_07_27_23_26_33_vs2012.sln	7/28/2015 11:29 AM	Microsoft Visual S...	2 KB
main_2015_07_27_23_26_33_vs2012.vcx...	7/28/2015 11:29 AM	VC++ Project	8 KB
main_2015_07_27_23_26_33_vs2013.open...	7/28/2015 3:42 PM	OPENSDF File	1 KB
main_2015_07_27_23_26_33_vs2013.sdf	7/28/2015 2:24 PM	SQL Server Comp...	18,048 KB
main_2015_07_27_23_26_33_vs2013.sln	7/28/2015 12:13 PM	Microsoft Visual S...	2 KB
main_2015_07_27_23_26_33_vs2013.v12....	7/28/2015 2:24 PM	Visual Studio Solu...	23 KB
main_2015_07_27_23_26_33_vs2013.vcx...	7/28/2015 3:42 PM	VC++ Project	9 KB
main_2015_07_27_23_26_33_vs2013.vcx...	7/28/2015 12:21 PM	Visual Studio Proj...	1 KB
NvEW.cpp	7/28/2015 11:29 AM	C++ Source file	140 KB
NvEW.h	7/28/2015 11:29 AM	C++ Header file	85 KB
PerfMarkersReset.cpp	7/28/2015 11:29 AM	C++ Source file	1 KB
PerfMarkersSetup.cpp	7/28/2015 11:29 AM	C++ Source file	1 KB
ReadOnlyDatabase.cpp	7/28/2015 11:29 AM	C++ Source file	20 KB
ReadOnlyDatabase.h	7/28/2015 11:29 AM	C++ Header file	10 KB
ReplayProcedures.cpp	7/28/2015 11:29 AM	C++ Source file	3 KB
ReplayProcedures.h	7/28/2015 11:29 AM	C++ Header file	1 KB

VS2010

VS2012

VS2013

Make sure you installed  
Nsight Tegra!



# COMPILE AND RUN

The screenshot displays the Microsoft Visual Studio IDE with the following components:

- Solution Explorer:** Shows a project named 'main\_2015\_07\_27\_23\_26\_33\_vs2013' with various source files like 'FramePart00.cpp', 'FrameSetup01.cpp', and 'FrameReset00.cpp'.
- Code Editor:** Displays the source code for 'RunFramePart00.NV\_EXEC'. The code includes OpenGL ES 2.0 calls for binding textures, setting uniforms, and drawing a range of elements. Line numbers 4722 through 4752 are visible.
- Output Window:** Shows the system output during the execution of the program on an NVIDIA SHIELD Tablet Android. It lists loaded system libraries such as 'libstagefright\_anc\_common.so', 'libstagefright\_enc\_common.so', and 'libstagefright\_omx.so'. The output concludes with the message: 'The program "[B485] com.nvidia.main\_2015\_07\_27\_23\_26\_33" has exited with code 1 (0x1)'.



# BENEFITS WITH GENERATED SOURCE

- You are denominating everything from bottom to top
  - Modify source and re-compile.
  - All CPU overhead is removed.
  - Add/remove/modify draw calls.
  - Change rendering state on the air.



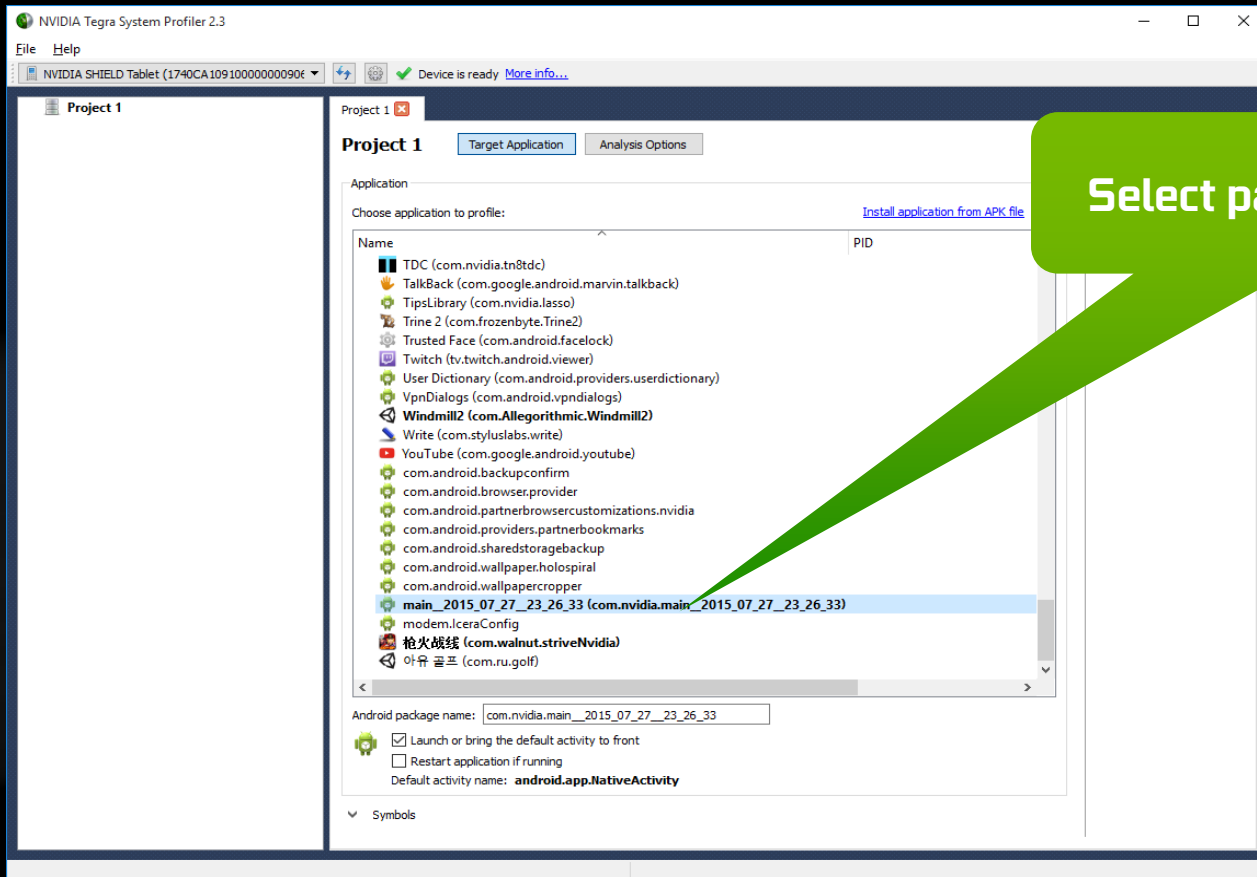


# TEGRA SYSTEM PROFILER

- Identify critical functions by sample count
- Call stack analysis
- Standalone tools on Windows/Linux/Mac.
- Support arm 32bit/64bit android



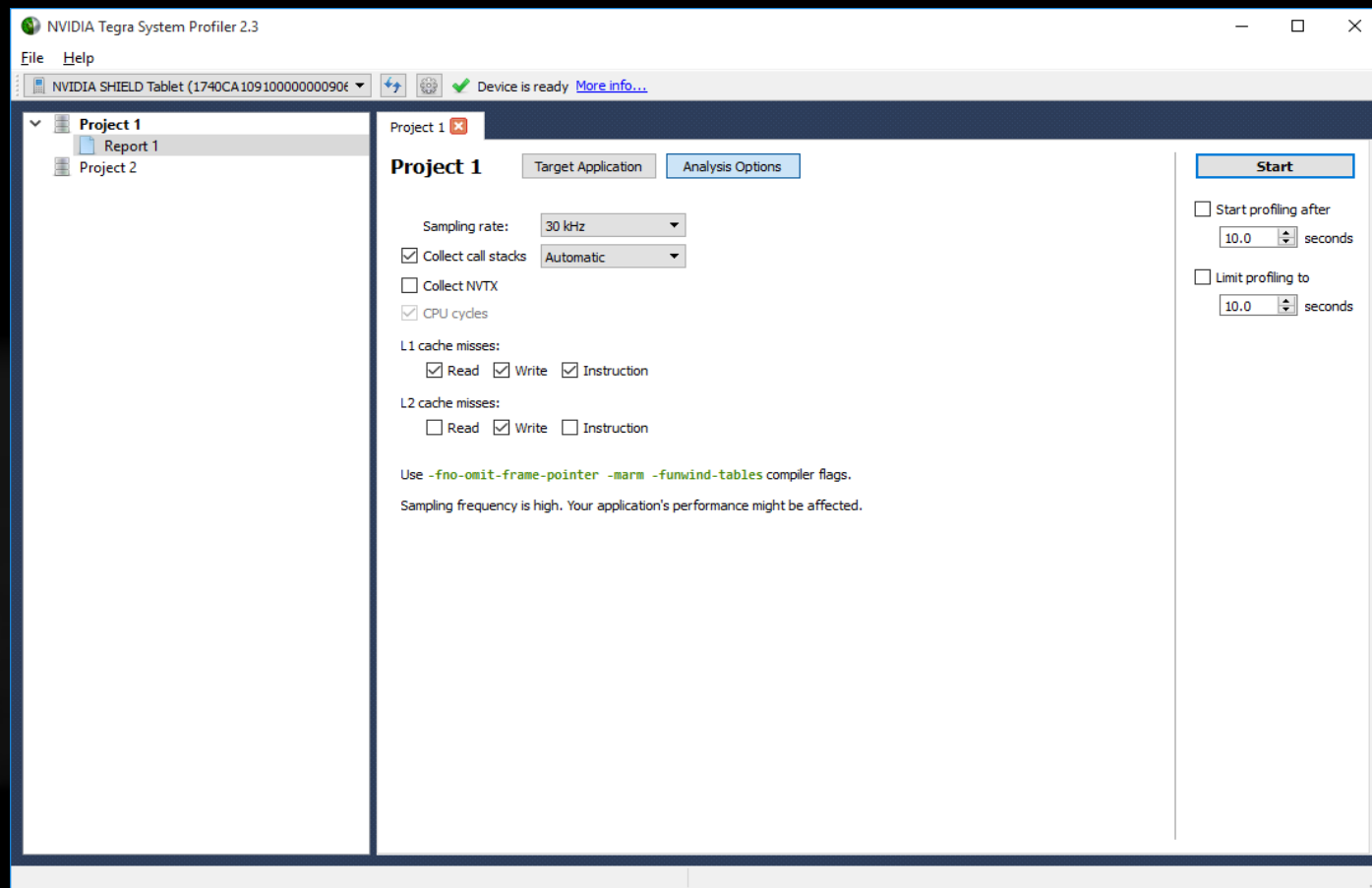
# TEGRA SYSTEM PROFILER



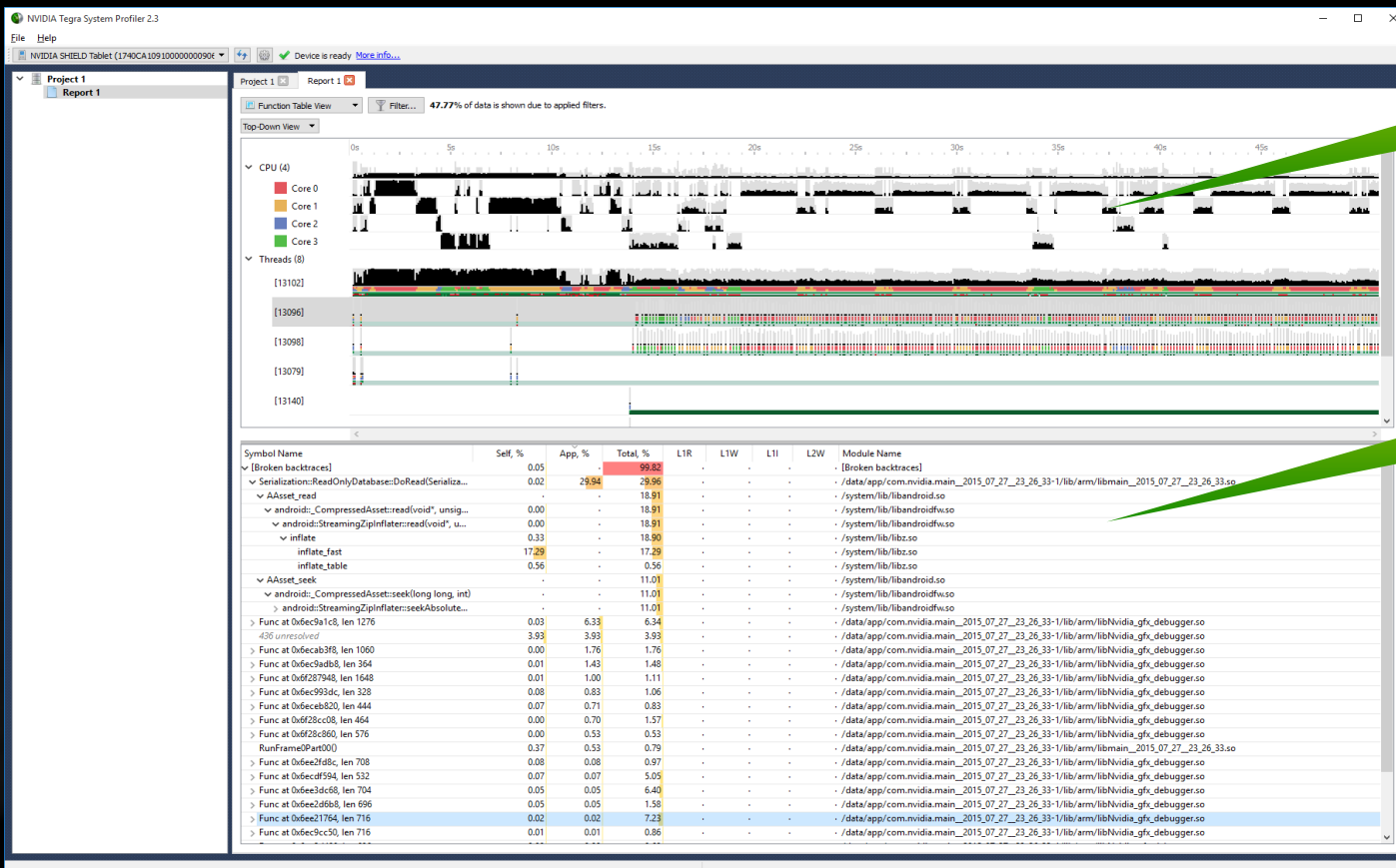
Select package to profile



# TEGRA SYSTEM PROFILER



# TEGRA SYSTEM PROFILER



CPU/Threads activity view

Function level profiling info



# NVIDIA GAMEWORKS™

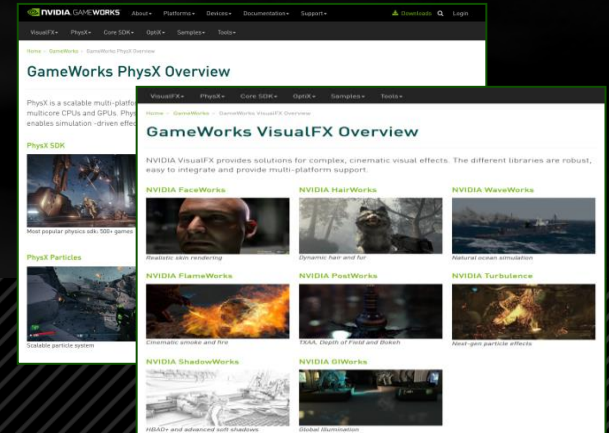
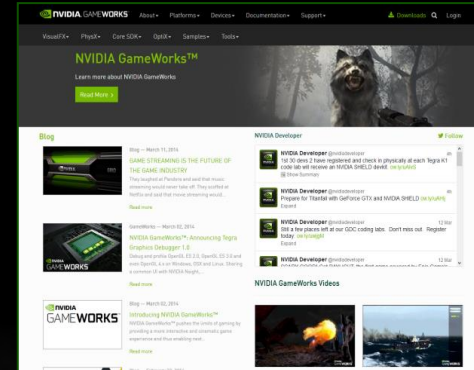
## Links...

Web: <http://gameworks.nvidia.com>  
Latest info and download options

YouTube: <https://www.youtube.com/user/nvidiaGameWorks>  
Tools, effects, and game integration videos

Twitter: <https://twitter.com/nvidiadeveloper>  
Catch up on up to the minute happenings

Survey: <https://developer.nvidia.com/developer-tools-survey-201503>



**THANK YOU**