

Spielekonsolenprogrammierung

Performance

- Profiling Strategie allgemein
- Übersicht über Razor
- Verwendung des Razor HUDs
- CPU Profiling
- GPU Profiling

- Profiling ist Vorbereitung des Tunings
- Tuning:
 - Optimierung der CPU Effizienz
 - Optimierung der GPU Effizienz
 - Optimierung der Speichereffizienz

- CPU Performance:
 - Sehr viel Zeit wird in kleinen Abschnitten gelassen (80/20 – Regel)
 - Unterscheide Implementierungs und Algorithmische Optimierung
 - Frage: Wo wird die Zeit liegen gelassen?
 - Ziel: Effizientes Arbeiten

Profiling Allgemein

- GPU – Profiling:
 - Werden Speichergrenzen überschritten?
 - Welches Teilsystem ist das limitierende (Geometrie, Rasterisierung, Speicher, ...)
- Profiler:
 - Selbstgeschriebene Profiler
 - Profiler – Werkzeuge (Itunes, SN Tuner)
 - Profiler – Werkzeuge mit eigener API

- Razor: Profiler Werkzeug mit eigener API
- 3 Komponenten:
 - Profiling HUD
 - CPU Profiler
 - GPU Profiler
- Relevante Doku:
 - libPerf in PSVita Library Doku
 - Performance Analysis and GPU Debugging in PDF Dokus

- Alle Razor Funktionen benötigen Module
 - SCE_SYSMODULE_RAZOR_HUD (hud)
 - SCE_SYSMODULE_PERF (CPU Analyse)
 - SCE_SYSMODULE_RAZOR_CAPTURE (GPU Analyse)
- Werden geladen mit:
 - sceSysmoduleLoadModule
- Libs:
 - SceSysmodule_stub (Für load Module)
 - SceRazorCapture_stub_weak

- Inbetriebnahme des HUDs:
- Include:
 - `#include <libsysmodule.h>`
 - `#include <razor_capture.h>`
- Modul laden bevor GXM initialisiert wird!
- Am besten alle Module laden, um CPU und GPU spezifische Informationen komplett zu erhalten
- Zeigt in GPU Buffer an, ob ein Überlauf vorliegt

- **Aufgabe 1:**
- Nehmen Sie das Razor HUD für ein beliebiges Programm in Betrieb
- CPU Performance im Tool geht nicht mit Razor HUD
- Eigener Menüeintrag Razor

- Wesentliche Ansicht
- Hierarchical Functions:
 - Zeigt an, welche Funktion am meisten kostet

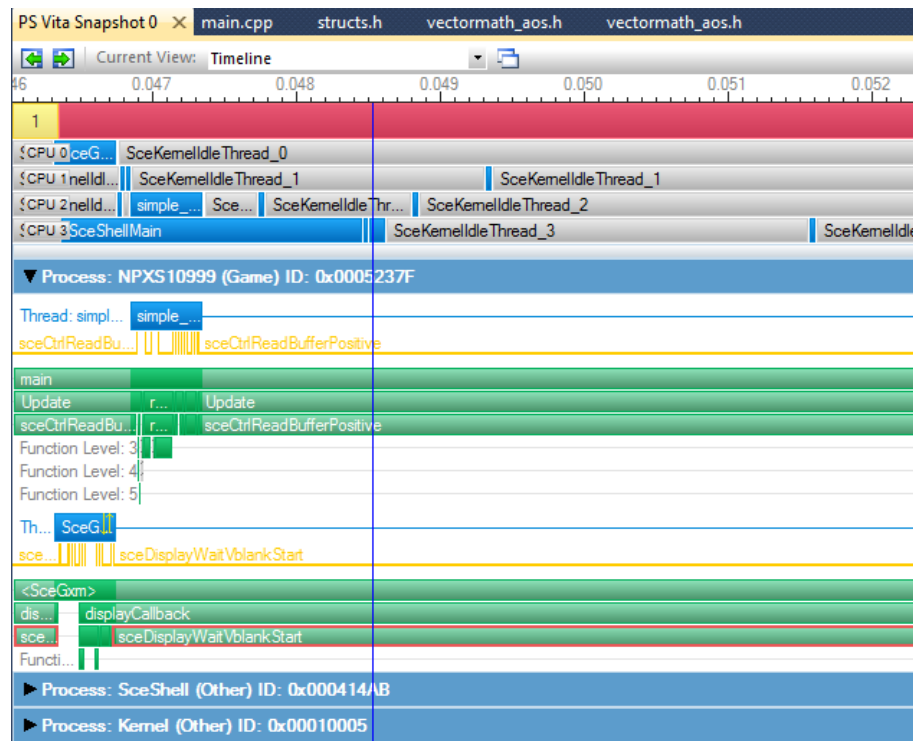
Current View: Hierarchical Functions

sceGxmDraw Show all threads

Name	Inc %	Exc %	Inc Time	Exc Time	Calls	Syscalls	Interrupts
sceDbgFontFlush	7.37 %	7.31 %	30.371 ms	30.120 ms	446	0	10
sceGxmEndScene	1.61 %	1.61 %	6.637 ms	6.637 ms	447	447	4
sceGxmDraw	0.95 %	0.95 %	3.898 ms	3.898 ms	894	0	5
sceGxmBeginScene	0.65 %	0.65 %	2.665 ms	2.665 ms	447	447	1
sce::Vectormath::Simd::Aos::Matrix4::operator* (sce::Vectormath::Simd::Aos::Vector4) const	0.67 %	0.56 %	2.760 ms	2.288 ms	3,576	0	4
Update()	3.86 %	0.32 %	15.927 ms	1.335 ms	448	0	1
sceGxmDisplayQueueAddEntry	0.31 %	0.31 %	1.281 ms	1.281 ms	447	894	0

Name	Inc %	Exc %	Inc Time	Exc Time	Calls	Syscalls	Interrupts
Processes	100.00 %	83.89 %	412.121 ms	345.722 ms	0	0	0
simple_main_thr (ID: 0x40010003)	51.85 %	43.63 %	213.671 ms	179.825 ms	0	0	0
SceGxmDisplayQueue (ID: 0x4001012B)	48.15 %	40.25 %	198.450 ms	165.897 ms	0	0	0

- Timeline gibt einen guten Überblick besonders über die Thread Belegung

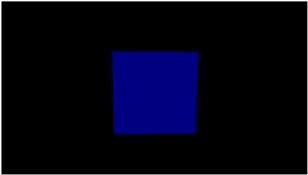


- **Aufgabe 2:**
 - Welche Funktion benötigt in ihrem Programm die meiste Zeit?
 - Genaueres Vermessen kann mit Hilfe der Funktionen
 - [scePerfArmPmonStart](#)
 - [scePerfArmPmonStop](#)
- ausgeführt werden.

- GPU Funktionen
- GPU Captcher dient zu Debugging Zwecken
- Gesamtübersicht

GPU Capture Info


GPU Capture




SELF	C:\Users\luerig\Documents\AltesDDrive\Le
File	Capture model 1
Scenes	1
Draws	2
Triangles	13
Date, Time	Montag, 18. August 2014 at 14:10:52
Captured By	luerig (from computer LUERIG-PC)

GPU Capture Views


API View



VDM Command Stream



Frame Resources View



- Was wird wie gezeichnet

```

sceGxmSetFrontVisibilityTestEnable(immCtx, SCE_GXM_VISIBILITY_TEST_DISABLED);
sceGxmSetBackVisibilityTestEnable(immCtx, SCE_GXM_VISIBILITY_TEST_DISABLED);
sceGxmSetFragmentProgram(immCtx, Fragment Program 0);
sceGxmSetVertexProgram(immCtx, Vertex Program 0);
sceGxmSetVertexStream(immCtx, 0, Vertex Stream 0);
sceGxmDraw(immCtx, SCE_GXM_PRIMITIVE_TRIANGLES, SCE_GXM_INDEX_FORMAT_U16, Index List 0, 3);
Draw 1: // sceGxmDraw(immCtx, SCE_GXM_PRIMITIVE_TRIANGLES, SCE_GXM_INDEX_FORMAT_U16, Index List 1, 3);
sceGxmSetFragmentProgram(immCtx, Fragment Program 1);
sceGxmSetVertexProgram(immCtx, Vertex Program 1);
sceGxmReserveVertexDefaultUniformBuffer(immCtx, Default_VerTEX_Uniform_Buffer);
sceGxmSetVertexStream(immCtx, 0, Vertex Stream 1);
sceGxmDraw(immCtx, SCE_GXM_PRIMITIVE_TRIANGLES, SCE_GXM_INDEX_FORMAT_U16, Index List 1, 3);
Terminate

```

Name	Wert	Typ
Vertex Streams		
Resource	Vertex Stream 0 : Attribute 0	Attribute Count
Memory	Main (Uncached)	Address
Index Source	16BIT	Format
Element Count	3	USSE PA Register Index
Offset	0 Bytes	Stride
Row	aPosition.x	aPosition.y
0	-1.0000	-1.0000

- **Aufgabe 3:**
 - Analysieren Sie Ihre Ausgabe mit GPU Capture
- GPU Trace bietet weitere Analysemöglichkeiten
- GPU Replay aus Capture File bietet Möglichkeiten zu analysieren, was passiert ist.

- Profiling Strategie allgemein
- Übersicht über Razor
- Verwendung des Razor HUDs
- CPU Profiling
- GPU Profiling