NVIDIA Nsight™ VSE 3.0
Catzilla Engine Development in DirectX® 11 and OpenGL 4.2

Jeff Kiel - NVIDIA Corporation
Michal Szymczyk - CTO, Plastic
Michal Staniszewski - Creative Director, Plastic
Agenda

- Catzilla Demo/Benchmark
  - Developed by Plastic, Platige Image and ALLPlayer
  - OpenGL 4.2 and DirectX®
- NVIDIA Nsight Visual Studio Edition Overview
- Problems During Development & Nsight Helped
- Q&A
Motivation behind the project

- There was a chance to do it - European Union Competitiveness & Innovation Programme (CIP)
- Old engine was completely tailored for PS3 (Datura® development)
- We wanted to prepare new engine for next gen machines
Development timeline

- Engine development before preproduction started - 2 months
- Preproduction - 1 month
- Production - 2 months
- Post Production - 2 months, 6 months total
Features of the engine

- Post processing (HDR, adaptive luminance, DOF with Bokeh)
- Fur based on geometry shader generated fins
- Physically based lighting
- Engine completely integrated with Autodesk® Maya
OpenGL/DirectX® 11

- OpenGL needed because of WindowsXP support, possibility to port to MacOSX and Linux
- Still using OpenGL inside of Autodesk® Maya
- Possibility to compare performance between two APIs
- DirectX still faster?
Benchmark modes

- Physics - CPU based using PhysX® - test
- Fur (Geometry Shaders) test
- Fluid (GPU Memory Bandwidth) test
- Raymarching (GPU ALU) test
NVIDIA Nsight Visual Studio Edition
Visual Studio integrated development for GPU and CPU

Build  Debug  Profile
NVIDIA Nsight Visual Studio Edition
Supports Direct3D 9/11 and OpenGL 4.2

Frame Debugger
• HUD for scene scrubbing
• State inspection at each draw call
• Real time frame capture and replay
• Source code serialization for D3D9/11

HLSL and GLSL Shader Debugger
• Native GPU shader debugging and GPU memory views
• Complex conditional breakpoints and Pixel History
• Local, single GPU shader debugging

Frame Profiler
• Automatic GPU bottleneck determination
• Draw call and frame timings
• Direct3D Perf Markers and render state grouping/sorting

Application and System Trace
• Inspect Direct3D & OpenGL, CPU & GPU workloads
• Correlate threads, call stack, API calls, WDDM kernel queues and resulting GPU workloads
• Concurrent draw call execution and memory transfer trace
Problems During Development

#1 Broken Bokeh Filter

Use similar algorithm to what DICE showed at SIGGRAPH 2011

Bokeh pass has extra bright lines & chess board pattern
Problems During Development

#1 Broken Bokeh Filter

Bug repro using D3D source generation for captured frames...edit generated code to quickly try debugging ideas.
Problems During Development

#1 Broken Bokeh Filter

Pixel History helps narrow down the draw calls
Problems During Development

#1 Broken Bokeh Filter

Dynamic Shader Editing to test the bug fix
Problems During Development

#2 Redundant State Changes

Analysis Summary shows high API Call (8615.6) to Draw Call (1012) ratio.
Problems During Development

#2 Redundant State Changes

Look at Event List to help confirm redundant state calls
Problems During Development

#2 Redundant State Changes

API Inspector allows you to make sure that code edits to remove redundant calls didn’t break anything.
Problems During Development

#2 Redundant State Changes

Reduced call ratio resulted in average of 15% decrease in frame time.
Problems During Development

#3 glMapBufferRange too expensive

3000 calls to glMapBufferRange taking 62% of the frame time
Problems During Development

#3 glMapBufferRange too expensive

- Looked at other possible solutions like buffer pool with fences but hurt SLI performance
- Turned out it was bad flags...was passing (GL_MAP_WRITE_BIT | GL_MAP_INVALIDATE_BUFFER_BIT | GL_MAP_UNSYNCHRONIZED_BIT) but the invalidate caused the driver to make too many temporary buffers
Problems During Development

#3 glMapBufferRange too expensive

Fix the flags, let the driver manage the memory, and perf improved!
Problems During Development

#4 Bug in SW culling

Scrubbing through the scene & saw items showing in depth buffer that never impacted the scene...

Using SW based method based on presentation by Daniel Conlin from DICE at GDC 2011
Problems During Development
#4 Bug in SW culling

ZCull activity in the profiler confirms the objects are not going to show...
Problems During Development

#4 Bug in SW culling

Note new summary screen for OpenGL Frame Profiler. Bars show unit bottleneck values and blue boxes will have red outline for areas of interest.
Problems During Development

#4 Bug in SW culling

Found bug in the linear depth calculation, reduced scenes by 400-500 draw calls
Problems During Rendering

#5 Indepth look at Raymarch Test

Benchmark designed to be GPU bound...target process showing GPU busy 100% of the time...that’s good!
Problems During Rendering

#5 Indepth look at Raymarch Test

Main thread active all of the time...repeating frame pattern

NVTX used to annotate the frame to help see what was happening in the scene
Problems During Rendering

#5 Indepth look at Raymarch Test

Row tooltips show per frame stats, Draw Call row shows concurrent draw calls

CPU is 2 frames behind the GPU, as expected for GPU bound app
Problems During Rendering

#5 Indepth look at Raymarch Test

3.0 includes WDDM tracing, green bars are packets with swap/present calls
Wrapping Up

Questions/Comments?

Resources

Info: https://developer.nvidia.com

Forums: https://devtalk.nvidia.com

Downloads

http://www.nvidia.com/nsight

http://www.allbenchmark.com/download

GDC Exhibit Booth: #1602