



# Instancing Tests

Bryan Dudash  
bdudash@nvidia.com

---

14 February 2007

# Document Change History

<b>Version</b>	<b>Date</b>	<b>Responsible</b>	<b>Reason for Change</b>
1.0	3/14/08	Bryan Dudash	Initial release

# Abstract

Instancing is an important tool for direct developers to help squeeze extra performance out of their complex 3D scene. It makes use of special hardware and low level driver code to reduce the overhead in drawing multiple copies of the same vertex buffer. In DirectX 10, the concept of instancing is built into the standard draw call.

InstancingTests provides a simple toolkit to performance analysis of instancing for various meshes and on various graphics chips.

## Motivation

Our goal with this sample is to provide a easy to use tool for benchmarking various instancing techniques against straight draw calls. It is not complicated, and can be customized by the user to run different meshes and/or shader code.

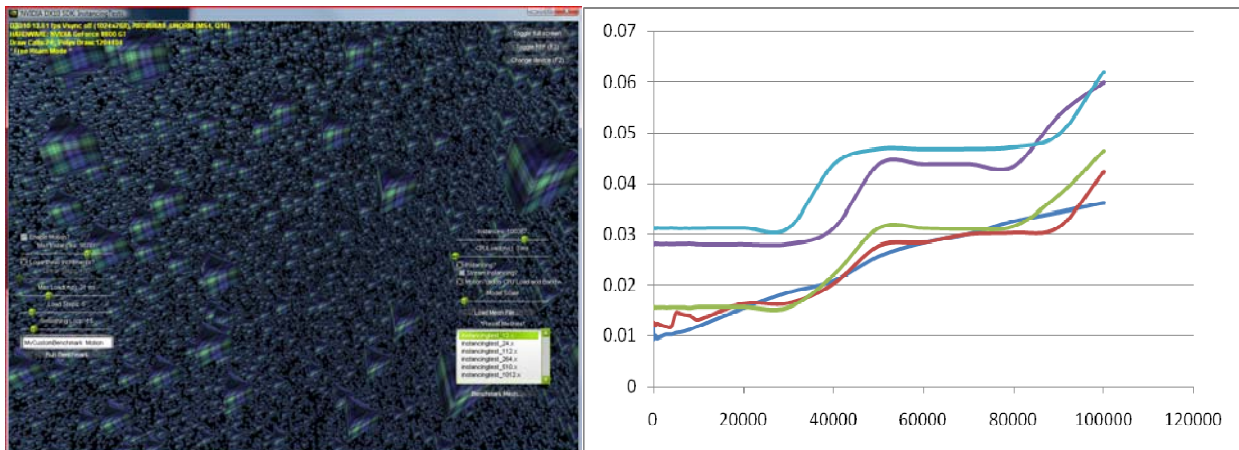


Figure 1:

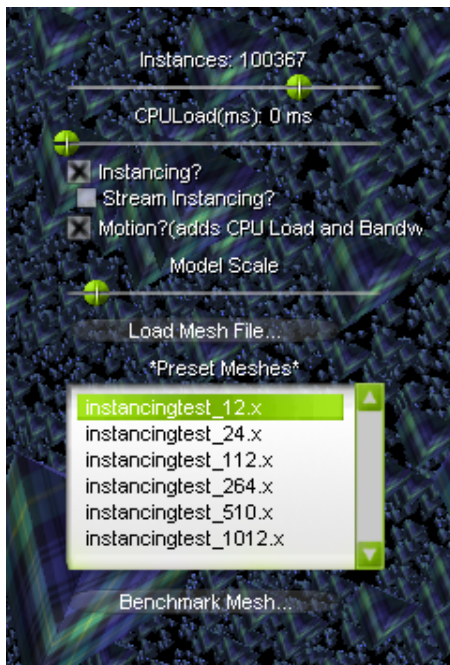
Left: Screen shot of InstancingTests in action.

Right: Graph generated in Microsoft Excel from CSV data dumped from InstancingTests. This particular graph shows the behavior of various CPU load levels on ConstantsBased Instancing of a 12 poly mesh as the # of instances varies. The Y axis is frametime. This graphic shows some interesting spikes as instances increase when under CPU load. Straight blue line is under no load.

# Usage Guide

This whitepaper will function more like a user guide. The concepts behind instancing are better covered in other texts.

The interface of InstancingTests contains two separate and distinct UI areas. On the right is the “Interactive” UI. This UI is active when the user is dynamically moving the camera and changing various options about the scene. The UIs are shown below.



Option	Description	Notes
<b>Instances</b>	The number of meshes to render	Upper limit based on multiples of how many instances can fit in a constant buffer.
<b>CPU Load</b>	Add artificial CPU load in milliseconds	Busy loop until the time is passed. Does not sleep.
<b>Instancing?</b>	Use instancing or not	Off means single draw calls.
<b>Stream Instancing?</b>	This enables use of “traditional” vertex stream based instancing	Off means constants based instancing
<b>Motion?</b>	Simple random oscillation of all meshes	Forces instance data buffer to update every frame
<b>Load Mesh File...</b>	Specify your own .x mesh to load	Opens a file dialog
<b>Preset Meshes</b>	A handy set of sphere meshes.	The number after the “_” indicates the poly count.
<b>Benchmark Mesh...</b>	Open the benchmark UI	Uses the currently loaded mesh.



Option	Description	Notes
<b>Enable Motion</b>	Set motion on or off for the benchmark	
<b>Max Instances</b>	Max for this benchmark run	Max capped by global max
<b>Logarithmic Increments</b>	10 steps from each factor of 10.	1,2...9,10,20...90,100,200... etc.
<b>Linear Steps</b>	Break range from 1 to max into a # of steps	Only used when not logarithmic
<b>Max Load(ms)</b>	Load is also tested across a range	Always linear
<b>Load Steps:</b>	Break load range into # steps	Capped by max load
<b>Smoothing Loop</b>	# of frames to average over	Loop count
<b>Benchmark (EditBox)</b>	Custom Benchmark Tag	Inserted into the output filename
<b>Run Benchmark</b>	Execute with above parameters	When finished results saved to a file in CWD.

**\*Note\***

The output filename with the results is generated to be of the following form  
*InstancingTests\_<BenchmarkTag>\_i<Max Instances>\_l<Max Load>.csv*

## DirectX 10's Draw overhead

This tool is designed to allow the user to get an idea of instancing performance gains and to do performance analysis. However, it is likely that there are some common results that are worth discussing. Microsoft has removed a lot of runtime verification of draw calls, and as a result the per-call overhead of Draw\*() under DirectX 10 is considerably lower than under previous DirectX versions. Nevertheless, there are still cases where using instancing will provide a significant performance due to reduction of CPU load. In addition, given the fact that GPU power is increasing faster than CPU power, the benefits of reducing CPU load will grow. The bottom line is that because each Draw call has some overhead, reducing the number of Draw calls will always increase efficiency.

**Notice**

ALL NVIDIA DESIGN SPECIFICATIONS, REFERENCE BOARDS, FILES, DRAWINGS, DIAGNOSTICS, LISTS, AND OTHER DOCUMENTS (TOGETHER AND SEPARATELY, "MATERIALS") ARE BEING PROVIDED "AS IS." NVIDIA MAKES NO WARRANTIES, EXPRESSED, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO THE MATERIALS, AND EXPRESSLY DISCLAIMS ALL IMPLIED WARRANTIES OF NONINFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A PARTICULAR PURPOSE.

Information furnished is believed to be accurate and reliable. However, NVIDIA Corporation assumes no responsibility for the consequences of use of such information or for any infringement of patents or other rights of third parties that may result from its use. No license is granted by implication or otherwise under any patent or patent rights of NVIDIA Corporation. Specifications mentioned in this publication are subject to change without notice. This publication supersedes and replaces all information previously supplied. NVIDIA Corporation products are not authorized for use as critical components in life support devices or systems without express written approval of NVIDIA Corporation.

**Trademarks**

NVIDIA, the NVIDIA logo, GeForce, and NVIDIA Quadro are trademarks or registered trademarks of NVIDIA Corporation in the United States and other countries. Other company and product names may be trademarks of the respective companies with which they are associated.

**Copyright**

© 2007 NVIDIA Corporation. All rights reserved.

**NVIDIA.**

NVIDIA Corporation  
2701 San Tomas Expressway  
Santa Clara, CA 95050  
[www.nvidia.com](http://www.nvidia.com)