ShaderBenchTool User Documentation

28th June 2004

What is ShaderBenchTool?

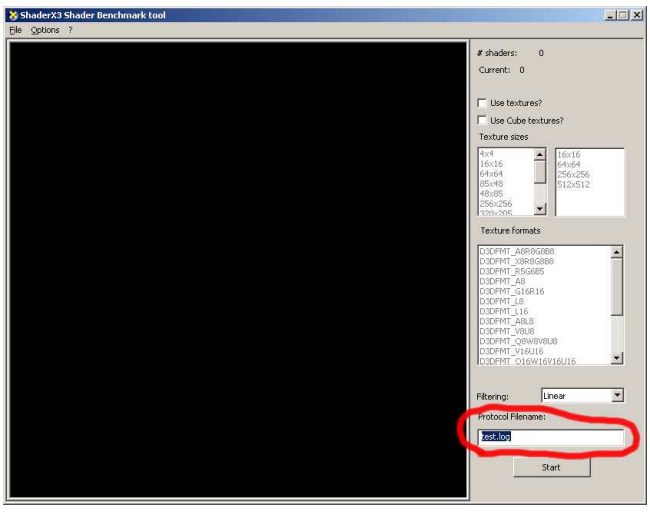
ShaderBenchTool is a tool to measure the pixel shader and texture processing performance of a graphics card and can therefore be used to evaluate the efficiency of a particular pixel program on the installed graphics adapter. It helps shader programmers to decide on optimization strategies by distinguishing between different variations and versions of a particular shader code through simply looking at the strain it produces. Additionally, it allows curious users to have a more in-detail look at the fragment processing performance of a particular video card.

How does ShaderBenchTool Work?

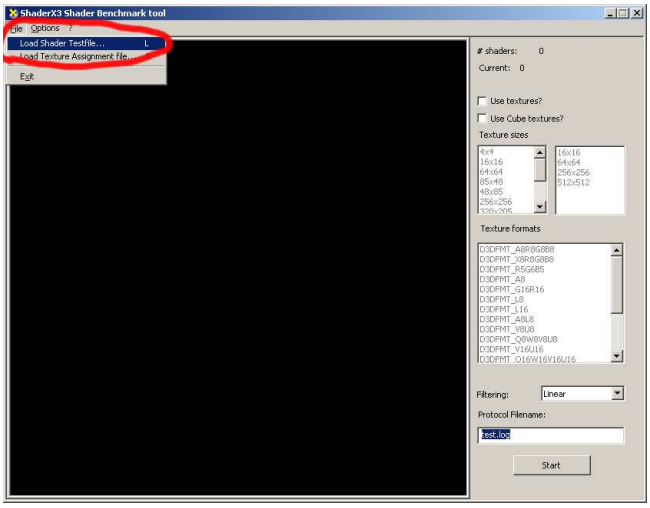
For getting meaningful measurements ShaderBenchTool produces a massive amount of overdraw to stress the fragment processing unit of the GPU. **Therefore, you should be extra careful when running this tool under Windows 2000 and Windows XP having a slow video card and very long and complicated shader programs as it might result in a hang-up or reboot of your machine.** This is due to a special build-in function in these two operating systems. This function normally is responsible for shuting down misbehaving device drivers that do not return control after a certain time period. The author(s) of the program do not provide any discharge to things that might happen to your hardware or software when this behavior occurs.

How do I use ShaderBenchTool?

First simply start the executable. After the application started and the main window showed up enter the filename of the benchmark protocol. In the default case it is a file named test.log located in the same directory

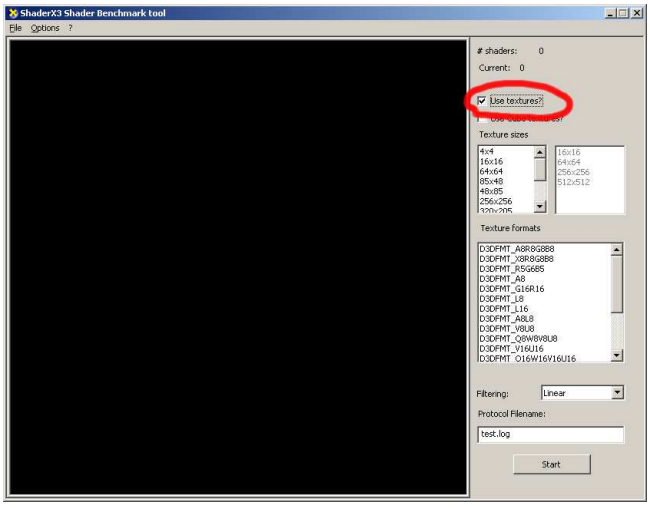


you started ShaderBenchTool from. Next load a shader testfile list by choosing the File menu and select “Load Shader Testfile”. A shader testfile contains all the filenames of the individual shaders you want to test in this run stored in ASCII text format. The following represents an example:



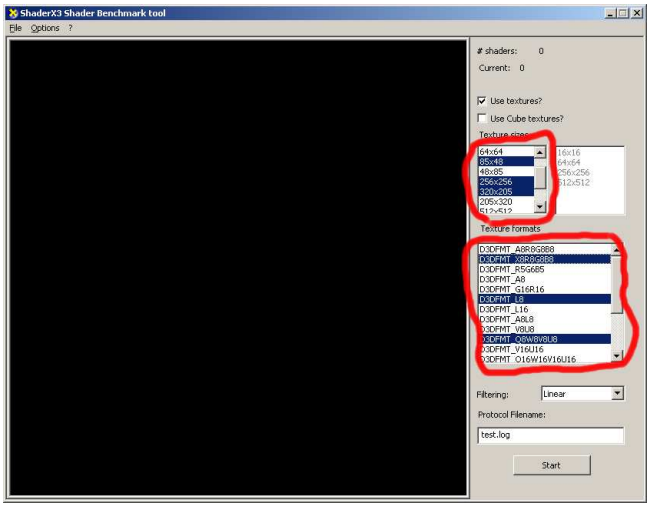

```
\shader\add.psh
\shader\mul.psh
\shader\my_fancy_shader.psh
\shader\my_fancy_shader_v2.psh
```

This shader testfile tells the ShaderBenchTool to test four different shader codes, *add.psh*, *mul.psh*, *my_fancy_shader.psh*, and *my_fancy_shader_v2.psh* exactly in this order. The filenames are specified relative to the directory where ShaderBenchTool is installed in. Therefore, it is wise to copy all the shaders you want to test in one particular directory like the $\backslash shader$ subdirectory. In this directory you can already find many example shaders. If your shaders contain texture load instructions you should click on the “Use textures”

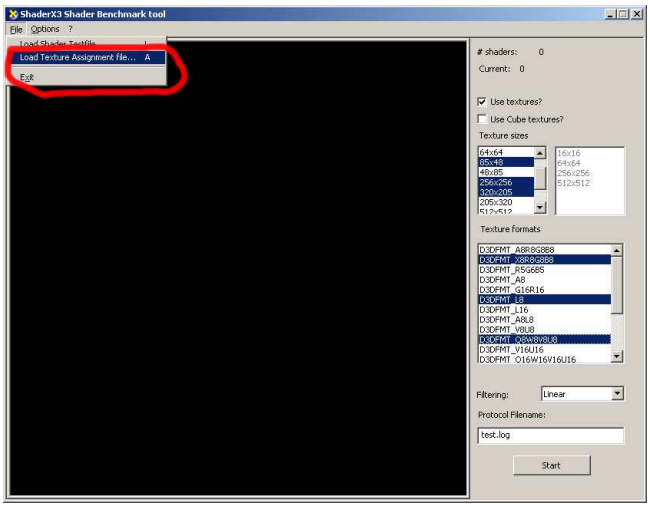


checkbox as seen on the picture above. If your shaders require cube texture maps activate the checkbox below. When the “Use textures” checkbox is enabled the lists for different texture formats and sizes will become activated. Now choose the sizes and texture formats you want to test your shaders with. **Again be careful, large texture sizes and texture formats with a high bit bandwidth may result in very long execution times on the card**. The texture format list contains only the texture formats that are currently supported by the graphics card according to the Direct3D driver. You can also choose between different texture sampling, e.g. linear or point sampling. Anisotropic filtering is not supported right now, because with the current measurement method anisotropic sampling does not have any impact on the performance. For the textures random patterns are generated and used for all texture samplers unless specified otherwise. If you want to have non-random textures but actual production textures you can load a texture assignment file to assign a texture to each sampler. The texture assignment file is an ASCII file just containing the absolute filenames of each texture you want to assign to a texture sampler. Here is an example

```
d:\fancy_artwork\fancy_artwork1.bmp
```




```
d:\fancy_artwork\fancy_artwork2.bmp
d:\fancy_artwork\fancy_artwork3.bmp
```



This assigns *fancy_artwork1.bmp* to sampler no. 0, *fancy_artwork2.bmp* to sampler no. 1 and so on. If you do not specify a texture for the remaining samplers, they will automatically use the last texture in the file. In our example case sampler no. 3 to 7 will be assigned to *fancy_artwork3.bmp*. When all necessary information are set, just press on the start button to execute the benchmark.

Comments,Questions, Bugs, Suggestions?

Feel free to drop us any message if you have comments, questions, found some bugs, or suggestions for improvements. Just send an e-mail either to diepstraten@vis.uni-stuttgart.de or mike@eissele.net.