NVIDIA®

OpenGL Multisample

Sébastien Dominé

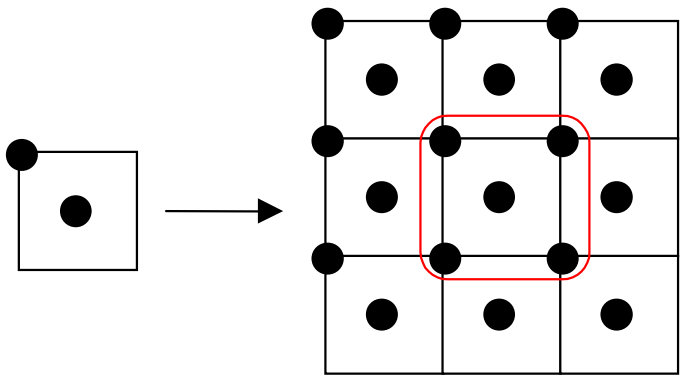
Agenda

- What is multisample?
- WGL extensions? WGL_ARB_extensions_string
- New way to query pixelformat?
  - WGL_ARB_pixel_format
- How to query for a multisample pixelformat?
  - WGL_ARB_multisample
- How to enable multisample in the application?
  - GL_ARB_multisample
- How to obtain a specific kernel filter on NVIDIA cards?
  - GL_NV_multisample_filter_hint
- Conclusion

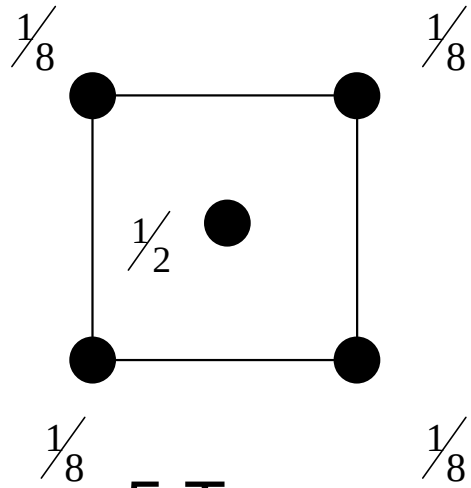
Getting started...

- Based upon WGL_ARB_multisample
- Requires WGL_ARB_extensions_string and WGL_ARB_pixel_format
- Allows the application to get a multisampled frame buffer with a given number of samples per pixels
- Multisample filtering taxonomy:
  - Sample : a subpixel frame buffer sample containing color, depth, and stencil information
  - Tap : source of data for filtering
- Not the same as Supersampling – no notion of Tap

Multisample pixel



2 Samples



5 Taps

2 Sample
5 Tap
Quincunx
Multi-Sampling

NVIDIA

Multisample frame buffer

- **It doesn’t happen for free**
- **More video memory is required:**

```
Vid_mem =
  sizeof(Front_buffer) +
  sizeof(Back_buffer) +
  num_samples * (sizeof(Front_buffer) +sizeof(ZS_buffer))
```

How does it happen in OpenGL

- WGL_ARB_extensions_string
- WGL_ARB_pixel_format
- WGL_ARB_multisample
- GL_ARB_multisample
- GL_NV_multisample_filter_hint

WGL extension string

- WGL_ARB_extensions_string
  - Don’t search the GL string for it – it is a WGL extension!
  - Query for the entry point:

    wglGetEntensionsStringARB = wglGetProcAddress(“wglGetEntensionsStringARB”)

    wglGetEntensionsStringARB != NULL

- const char * wgl_ext_string = wglGetEntensionsStringARB(hDC);

New way to query pixel formats - Trick

- Multisample window in OpenGL:
  - WGL_ARB_pixel_format and WGL_ARB_multisample extensions
  - Need to create a dummy window to collect the extensions and entry points bound to a hardware accelerated context
  - Then you can create the multisample window

New way to query pixel formats

- Win32 GL function
  - ChoosePixelFormat, DescribePixelFormat are not extensible
- WGL_ARB_pixel_format
  - Array of paired attribute/value to describe a pixel format
  - Adds wglChoosePixelFormat and wglGetPixelFormatAttrib{fi}vARB – still uses SetPixelFormat
  - Well defined matching comparison function per attribute

WGL ARB pixel format

```
BOOL wglChoosePixelFormatARB( HDC hdc,
                              const int *piAttribIList,
                              const FLOAT *pfAttribFList,
                              UINT nMaxFormats,
                              int *piFormats,
                              UINT *nNumFormats);
float fAttributes[] = {0, 0};
int   pixelFormat;
UINT  numFormats;
int   iAttributes[] = { WGL_DOUBLE_BUFFER_ARB, TRUE,
         WGL_ACCELERATION_ARB,  WGL_FULL_ACCELERATION_ARB,
         0, 0 };
status = wglChoosePixelFormat(hDC, iAttributes, fAttributes,
                        1, &pixelFormat, &numFormats);
```

WGL ARB pixel format - Trick

- **If wglChoosePixelFormat succeeds, it doesn't mean it found a matching PFD – need to test numFormats**

WGL ARB pixel format

```
BOOL wglGetPixelFormatAttribivARB( HDC hdc,
                                   int iPixelFormat,
                                   int iLayerPlane,
                                   UINT nAttributes,
                                   const int *piAttributes,
                                   int *piValues);
```

```
int iAttributes[3];
int iResults[3];
iAttributes[0] = WGL_DOUBLE_BUFFER_ARB;
iAttributes[1] = WGL_ACCELERATION_ARB;
iAttributes[2] = 0;
status = wglGetPixelFormatAttribivARB(hDC, pxlfmt, 0, 2,
                                      iAttributes, iResults);
```

Query for a multisample pixelformat

- WGL_ARB_multisample
  - Add the following pixel format attributes:
    - WGL_SAMPLE_BUFFERS_ARB
    - WGL_SAMPLES_ARB

```
iAttributes[0] = WGL_SAMPLE_BUFFERS_ARB;
iAttributes[1] = 1;
iAttributes[2] = WGL_SAMPLES_ARB;
iAttributes[3] = 2; // could be 4
```

And call wglChoosePixelFormat...

Enable multisample

- We now have a multisample window
  - Enabling/Disabling multisample rasterization:

```
glEnable(GL_MULTISAMPLE_ARB)
glDisable(GL_MULTISAMPLE_ARB)
```

GL_ARB_multisample

- **Multisample Fragment Operations:**
  - **Explicit Fragment Coverage Value**

    void glSampleCoverageARB(GLclampf *value*, GLboolean *invert*)

  - **Assign Fragment Coverage Value based on Fragment Alpha**
    - **Sub pixel Screen door transparency**
  - **Force Fragment Alpha Value to 1**

GL_NV_multisample_filter_hint

- NVIDIA extension explicitly exposes the different filters:

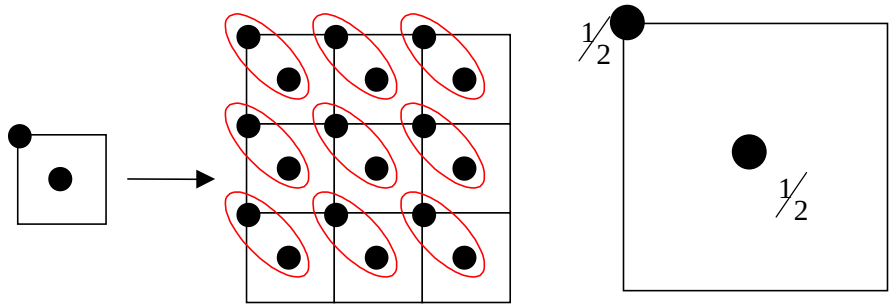
```
glHint(GL_MULTISAMPLE_FILTER_HINT_NV, GL_FASTEST);
glHint(GL_MULTISAMPLE_FILTER_HINT_NV, GL_NICEST);
```

GeForce 3/4Ti:

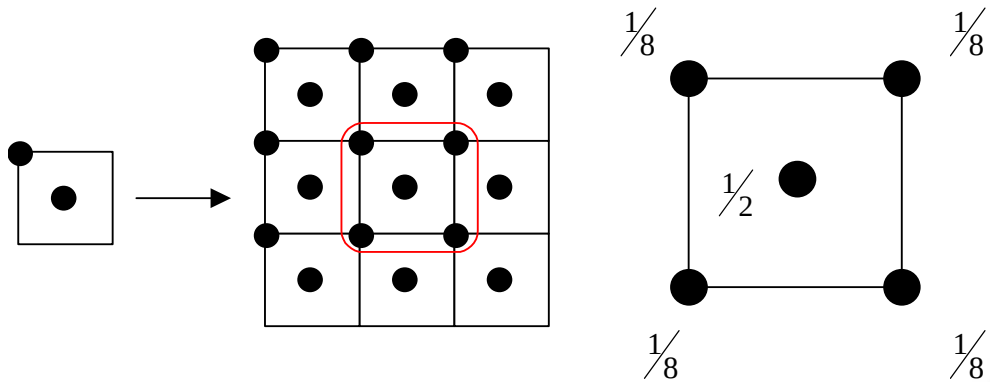
Samples	FASTEST	NICEST
2	2 Samples 2 Taps Box	2 Samples 5 Taps Quincunx
4	4 Samples 4 Taps Box	4 Samples 9 Taps Box

GL_NV_multisample_filter_hint

2 Sample
2 Tap
Box
Multi-Sampling



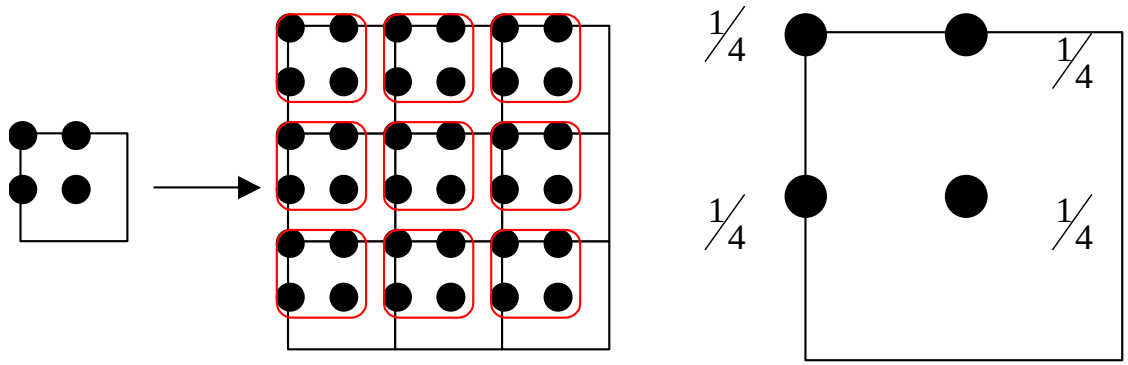
2 Sample
5 Tap
Quincunx
Multi-Sampling



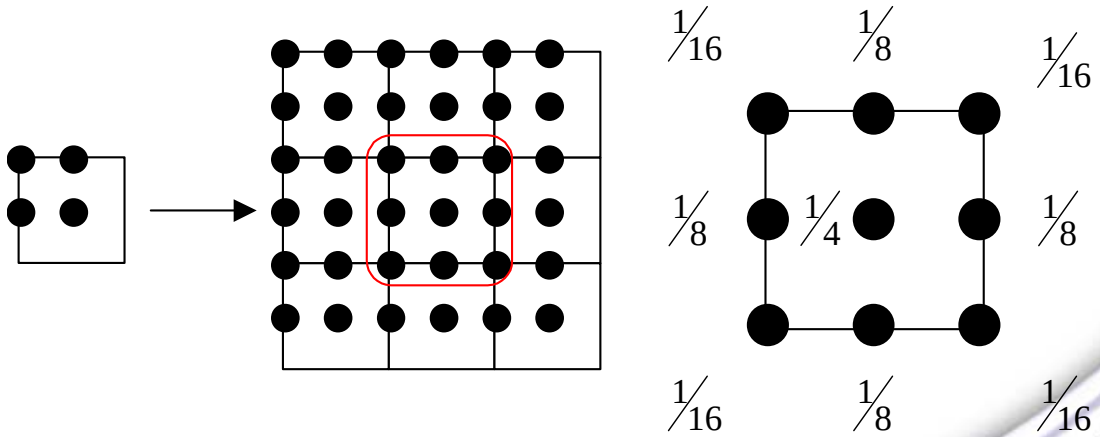
NVIDIA.

GL_NV_multisample_filter_hint

4 Sample
4 Tap
Box
Multi-Sampling



4 Sample
9 Tap
Box
Multi-Sampling



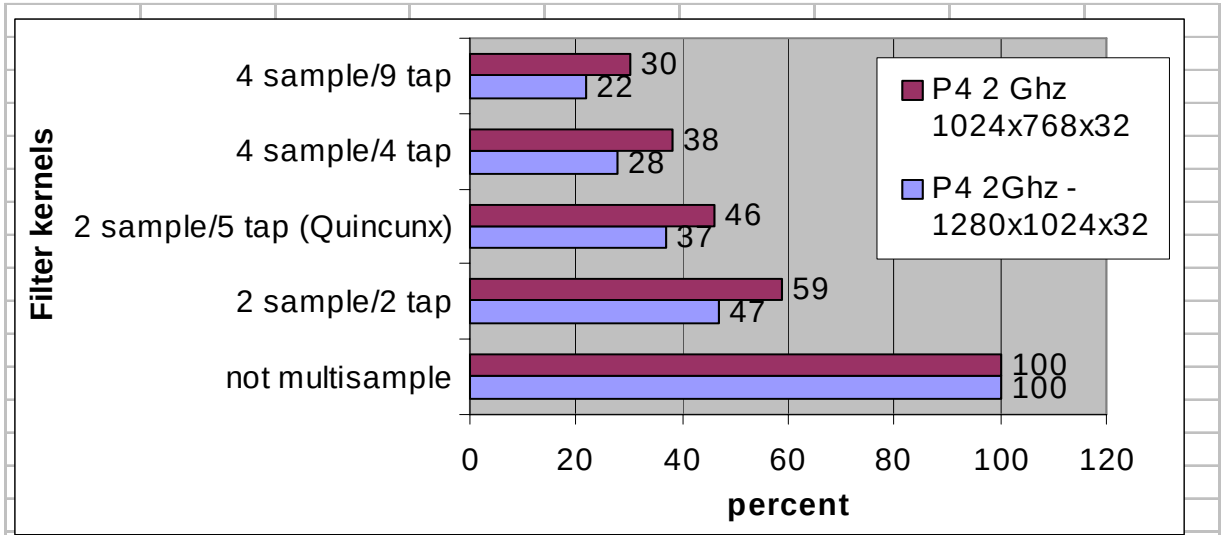
NVIDIA.

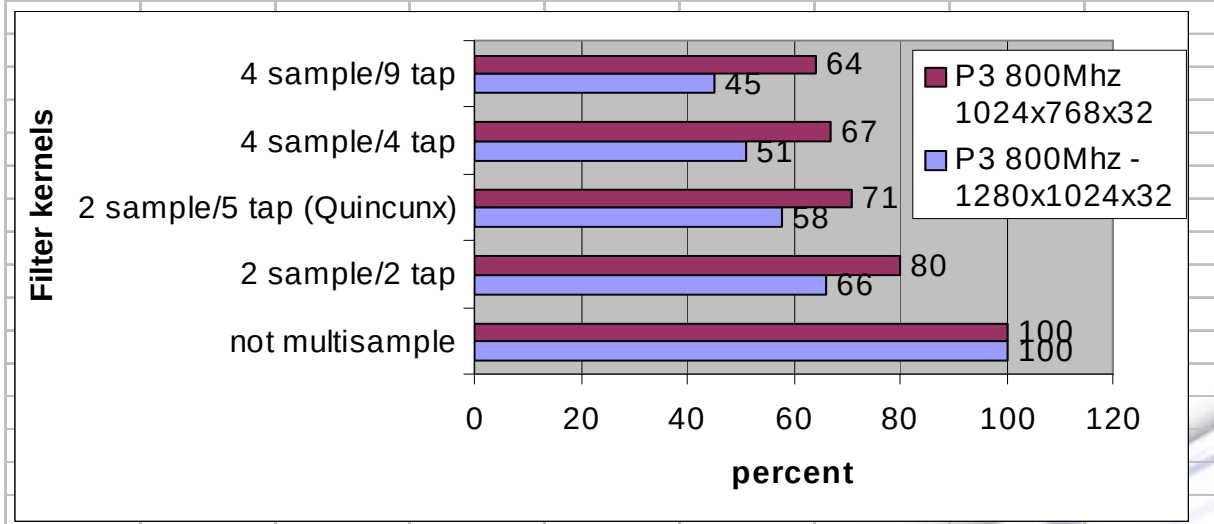
GL_NV_multisample_filter_hint

Performance considerations

- Texture thrashing?
- If T&L or CPU bound – Multisample can be cheap
- Scalability forces the video options to enumerate all the modes
- Test case – Quake III

Performance considerations





NVIDIA

Conclusion

- Easy to support
- Tremendous visual quality improvement
- Explicit control of the what should or shouldn’t be antialiased
- Doesn’t necessarily mean slowdown
- Much nicer than a Display Driver control panel toggle

- Demo
  - NV Pixel Format 1.0