

Implement Off-Screen MSAA


| Revision | Author | Date | Reviewer | Approver | Disapproval | Changelog | JIRA Story |
|-----------|-----------------|-------------|--|--|--|-------------------------------|---|
| "rev 0.1" | Zhang, Mingming | 30 Jun 2022 | <input type="checkbox"/> Adrian Tut <input type="checkbox"/> Zhang, Xufeng <input type="checkbox"/> Laurentiu Pinte <input type="checkbox"/> Mirela Kiss <input type="checkbox"/> Razvan Oprea <input type="checkbox"/> Csaba Abram <input type="checkbox"/> Marius Pop <input type="checkbox"/> Andrii Udovenko - (p) <input type="checkbox"/> Tamas Both <input type="checkbox"/> Lu, Wenting | <input type="checkbox"/> Adrian Tut <input type="checkbox"/> Zhang, Xufeng <input type="checkbox"/> Laurentiu Pinte <input type="checkbox"/> Mirela Kiss <input type="checkbox"/> Razvan Oprea <input type="checkbox"/> Csaba Abram <input type="checkbox"/> Marius Pop <input type="checkbox"/> Andrii Udovenko - (p) <input type="checkbox"/> Tamas Both <input type="checkbox"/> Lu, Wenting | <input type="checkbox"/> Adrian Tut <input type="checkbox"/> Zhang, Xufeng <input type="checkbox"/> Laurentiu Pinte <input type="checkbox"/> Mirela Kiss <input type="checkbox"/> Razvan Oprea <input type="checkbox"/> Csaba Abram <input type="checkbox"/> Marius Pop <input type="checkbox"/> Andrii Udovenko - (p) <input type="checkbox"/> Tamas Both <input type="checkbox"/> Lu, Wenting | Implement the off-screen MSAA |  TASDK-24600 - Investigate and Implement the MSAA on Atlas engine CODE REVIEW |

Table of Contents

- [Introduction](#)
- [Requirements](#)
- [Acceptance Criteria](#)
- [Design](#)
 - [Solution 1](#)
 - [Solution 2](#)
 - [Class Diagram](#)
 - [MSAA Rendeing Pipeline](#)
- [User Interfaces](#)
 - [Configuration](#)
- [Change Impact](#)
 - [Performance and Memory Impact](#)
 - [Test Images](#)
 - [Backward Compatibility Impact](#)
- [Others \(FAQ, References, ...\)](#)

Introduction

As a user, I would like to see smooth, non-aliased elements on the map (roads, buildings, etc) so that I can have a delightful experience when exploring the map.

Currently, the buildings and roads have an aliasing effect, we need to try different anti-alias technology to reduce it. even we've used FSAA, but its effect is not perfect.

Multisample anti-aliasing (MSAA) is a type of spatial anti-aliasing, a technique used in computer graphics to remove jaggies.

Here introduces it into Atlas engine as an option to improve picture quality and we could set 4x, 8x by configure.

Requirements

ANDROID-3010 Anti-aliasing using the MSAA technique

We've used FXAA for buildings on C++, and tried multiple other hack-fixes to try and hide the aliasing effects on the roads, but with no huge impact:

- <https://jira.telenav.com:8443/browse/TASDK-21062>
- <https://jira.telenav.com:8443/browse/TASDK-21949>

On Android, anti-aliasing could be better achieved using multi-sample anti-aliasing.

We need to start researching this option and decide what our course of action will be.

Acceptance Criteria

AC should focus on below 4 major aspects:

- *non-aliased elements on the map (Road, polygon, building, landmark, just the same scope of the current FXAA.*
- *Multiple samples should be configurable
It should be 4, 8, 16, when larger than the GPU maximum samples just using maximum GPU samples.*
- *no obvious performance degradation.*
- Support to open FXAA and MSAA at the same time

Design

~~Solution 1~~

Using **texelFetch + glTexStorage2DMultisample**

Advantage support to design your own sampling algorithm in shader, it will be more programmable.

Disadvantage use GLES 3.1 new feature glTexStorage2DMultisample <https://docs.gl/es3/glTexStorage2DMultisample>, but most of our client just support ES 3.0, in long term perspective, we should also support this.

Solution 2

Using **glBlitFramebuffer**

Advantage: glBlitFramebuffer need GLES 3.0 <https://docs.gl/es3/glBlitFramebuffer>, meet the most client's GL version, no shader changes, code is simple.

Disadvantage the processed texture can't be used(binded) directly, need to use "blit" to the target frame buffer; the sampling algorithm is fixed (OpenGL defined), not be programmable.

Consider that most of our client just support ES 3.0, currently, Implement the **solution 2** for atlas engine.

Class Diagram

API changes

```

class Framebuffer
{
    o samples: uint32_t
    o isMsaaf: bool
    e Framebuffer(uint32_t width, uint32_t height, bool withStencil = false, bool withDepth = false, bool isMsaaf = false, uint32_t samples = 0)
}

```

```

class OpenGLRenderTarget
{
    o m_msa_rbo_handle : GLuint
    e isMsaafRenderTarget() const : bool
    e copy(const OpenGLRenderTarget& src, const OpenGLRenderTarget* des = nullptr) : void
}

```

```

class IRenderInterface
{
    // Binds resolve the multiple sample RenderTarget to the normal render target
    ==
    // Enable of disable writing to the color buffer.
    e resolve_msaaf(const RenderTarget& msaaf_target, const RenderTarget* naomaf_target) : void
    e setColorWrite(bool red, bool green, bool blue, bool alpha) : void
}

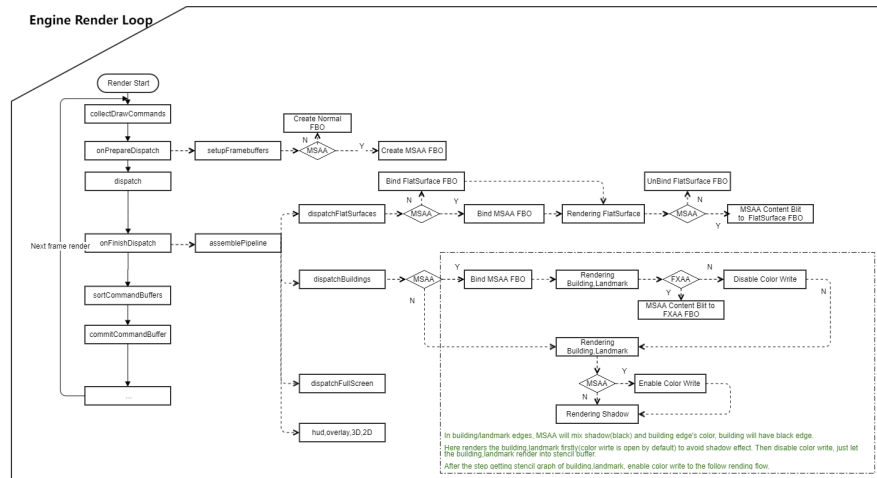
```

```

class OpenGLRenderInterface
{
    e resolve_msaaf(const gl::OpenGLRenderTarget& msaaf_target, const gl::OpenGLRenderTarget* naomaf_target = nullptr) : void
    e setColorWrite(bool red, bool green, bool blue, bool alpha) : void
}

```

MSAA Rendeing Pipeline



```

view_settings
{
    msa
    {
        enabled    false
        samples    4
    }
}

```

Change Impact

Performance and Memory Impact

Test steps of Atlas performance: <https://spaces.telenav.com:8443/x/FYspDQ>

Environment: HP, win10, Intel(R) UHD Graphics 620

Data: NA20Q2

Test data **xlsx**: in the attachment of [TASDK-24600 - Investigate and Implement the MSAA on Atlas engine](#) CODE REVIEW named **benchmark.7z**.

Get avg from all **xlsx** files and calculate by hand

| 5s fps avg | msaa off, with 3D buiding | msaa off, without 3D buiding | msaa on 4x, with 3D buiding | msaa on 8x, with 3D buiding | msaa on 16x, with 3D buiding |
|------------|---------------------------|------------------------------|-----------------------------|-----------------------------|------------------------------|
| | 441.80 | 478.2226 | 277.90 | 206.17 | 127.25 |
| | 478.31 | 592.8634 | 263.51 | 206.70 | 144.55 |
| | 510.33 | 618.9639 | 287.84 | 209.16 | 135.13 |
| AVG | 476.8133 | 563.35 | 275.675 | 207.3433 | 135.6433 |
| % | 100% | 118.15% | 57.81% | 43.48% | 28.448% |
| | 0 | +18.15% | -42.18% | -56.51% | -71.55% |

MSAA 4x performance degradation **-42.18%**

MSAA 8x performance degradation **-56.51%**

MSAA 16x performance degradation **-71.55%**

Test reprot print by running the python script : https://bitbucket.telenav.com/projects/NAV/repos/tasdk-nav-core/browse/mapdisplay2/tools/benchmark/process_kpi.py

<https://jira.telenav.com:8443/secure/attachment/762865/report.txt>

| | msaa off, with 3D buiding | msaa off, without 3D buiding | msaa on 4x, with 3D buiding | msaa on 8x, with 3D buiding | msaa on 16x, with 3D buiding |
|------------|---------------------------|------------------------------|-----------------------------|-----------------------------|------------------------------|
| | 379.650449 | 395.802080 | 293.585133 | 227.297507 | 149.894765 |
| | 403.024532 | 464.147817 | 275.051712 | 227.297507 | 158.841303 |
| | 407.566800 | 493.294706 | 278.868493 | 218.580432 | 153.795156 |
| AVG | 396.7473 | 451.0815 | 282.5018 | 224.3918 | 154.1771 |
| % | 100% | 113.695% | 71.204% | 56.558% | 38.86% |
| | 0 | +13.695% | -28.795% | -43.442% | -61.14% |

MSAA 4x performance degradation **-28.795%**

MSAA 8x performance degradation **-43.442%**

MSAA 16x performance degradation **-61.14%**

Generally, the MSAA is expensive [anti-aliasing method](#), suggest to use 4x, then 8x, not use 16x.

Test Images

Origin

MSAA x4

origin VS MSAA x4
left is origin, right the MSAA

MSAA x8

origin VS MSAA x8
left is origin, right the MSAA

FXAA + MSAA x8

FXAA

Backward Compatibility Impact

No impact on the backward compatibility, it's a new configure, no public API change.

Others (FAQ, References, ...)