리졸브를 해결하라
(Resolve Your Resolves)

홀거 그루엔/존 스토리(Holger Gruen and Jon Story)
가마수트라 등록일(2008. 3. 26)

http://www.gamasutra.com/view/feature/3591/resolve_your_resolves.php

[이번 심층 칼럼에서 그루엔(Holger Gruen)씨와 스토리(John Story)씨는 게임 개발의 안티 앨리어싱 (anti-aliasing)과 PC 게임에서 어떻게 재기스 (jaggies)를 감소 시킬수 있는지에 대해 논해보고자 한다. 또한 보다 적은 포스트 프로세싱(post processing) 패스(pass)를 이용함으로서 프레임 비율을 절감할수 있는 법에 대해서도 알아보도록 하다.]

지난 몇 년간 MSAA (Multi-Sample Anti- Aliasing) 을 사용하여 고화질 렌더링(rendering)을 이끌어내는 PC 게임들을 흔히 볼 수 있다.

MSAA 는 삼각 래스터화(rasterization) 과정에서 발생하는 보기 흉한 재기스의 감소에 매우 효과가 있는 방법이다. 또한 대부분의 게임 엔진들은 피사계 심도(depth of field), 모션블러 (motion blur/잔상), 색보정, 굴절 과 같은 포스트 프로세싱(post processing) 기술을 이용하기도 한다.

포스트 프로세싱은 점점 많은 인기를 끌고 있다. 복잡한 컴퓨테이션(computation)을 행하는 역할을 수행할뿐더러, 시각적으로 확인 가능한 픽셀(pixel) 구현에 대한 비용만이 들뿐이다. 사실 20 개까지의 패스(pass)를 포함하는 엔진은 알려지지 않았으며, 이런 기술들을 사용시에는 보통 주요 렌더 타겟(render target)으로 설정되어 그려진 이미지의 텍스쳐(texture) 파일을 텍스쳐 인풋(input)으로서 사용키 위해 복사하는 과정이 필요하다.

MSAA 를 사용하는 엔진에서의 렌더 타겟은 다음 패스에서 이용되기 전에서 리졸브(resolve) 되어야 한다. 그리고 이런 이미지를 생성하기 위해서는 어떤 버전의

D3D 를 사용하느냐에 따라 *IDirect3DDevice9::StretchRect* 또는 *ID3D10Device::ResolveSubresource* 의 함수를 호출하여 이뤄진다.

현대 게임 엔진들이 다수의 포스트 프로세싱 기술들을 적용하는 경향이 있기에, 리졸브의 루프(loop)를 어떻게 발생 시키는지 쉽게 알 수 있다. (그림 1.)

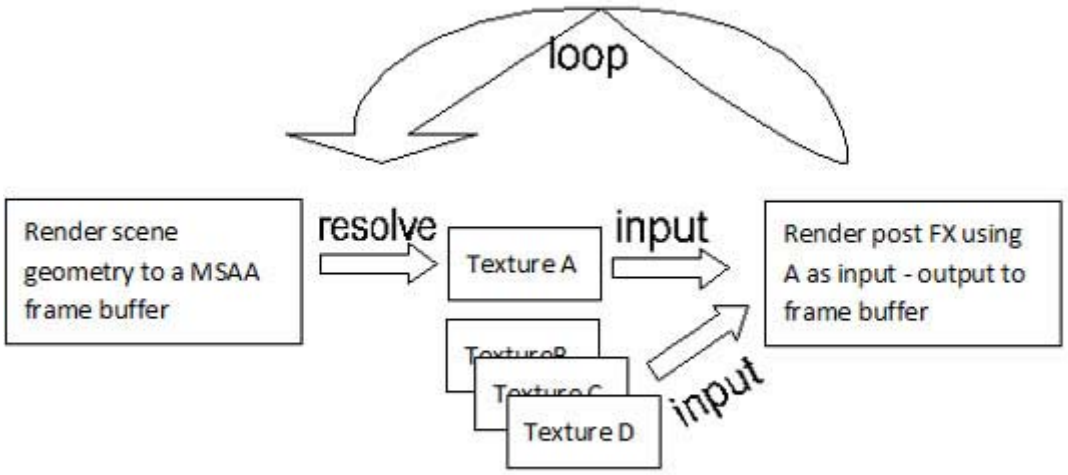

Figure 1

리졸브는 공짜로 이뤄지는 공정이 아니며, 프레임(frame) 당 다중 리졸브를 처리하는 것은 실행에 심각한 문제를 초래할 수 있다는 것을 명심해야 한다. 이는 모든 그래픽 하드웨어에 해당되는 사실일 것이다.

실제 예를 들어보자면, 최근 발매된 PC 게임의 개발자들은 리졸브 카운트(리졸브 카운트)를 경이적인 22 에서 12 로 낮추는데 성공하였다. 이로 인해 1280x1024@4xAA 의 해상도(resolution)로 프레임 당 약 12ms 의 단축을 이뤄냈다.

이번 칼럼의 목적은 포스트 프로세싱 효과나 지연 셰이딩(deferred shading)의 질을 손상시키지 않고 렌더링 파이프라인(rendering pipeline)에서의 리졸브 카운트를 최소한 하는 법에 대한 해설을 하고자 하는데 있다.

제거되어야 하는 리졸브은 중복리졸브와 유해리졸브 두 종류로 구분된다. 이에 대해선 후반부에서 자세히 설명키로 하고 일단 고화질 영상에 필요한 리졸브들에 대해 알아보도록 하자.

필요한 리졸브

그리기 요청 (draw call) 에 의해 시각으로 확인 가능한 재기스가 발생할 경우에만 MSAA 렌더 타겟들의 사용이 효과가 있다. 주요 지오메트리 패스 (geometry pass) 가 MSAA 형식으로 렌더링 한 후, 비 MSAA 렌더 타겟으로 리졸브하는 것이 이상적 일수 있다. 후에 일어나는 포스트 프로세싱 패스들은 모두 비 MSAA 형식으로 완성되어 짐으로서, 프레임 당 단일 리졸브을 발생시킬 수 있는 것이다.

그러나 포스트 프로세싱 기술이 MSAA 형식으로 처리되어야 하는 두 가지 이유가 있다.

1. 만약 포스트 프로세싱이 부표본(subsample)을 기반으로 한 깊이 검사 (depth testing) 를 작동시킨다면, 한 픽셀의 몇몇 부표본들이 업데이트 될 것이다.
2. 비슷한 방식으로, 알파 블렌딩(alpha blending)이 작동된다면 부표본 정보들은 그 블렌딩 작업 동안 저장되어지기 때문이다.

이 두 경우에선 물론 그 이상의 패스들에 대한 렌더 타겟을 리졸브하는 것이 맞을지도 모른다. 그러나 이 두 가지 경우는 예외일 뿐이며, 깊이 검사 (depth testing) 나 알파 블렌딩을 작동 시키지 않는 최대 화면의 패스들에 대해서는 MSAA 형식으로 처리할 필요가 없다는 것을 명심해야 한다.

중복리졸브

4 분활 최대화면만을 나타내는 기술은 MSAA 렌더 타겟의 모든 부표본들을 같은 색깔로만 보통 인식을 한다. (Figure 2)

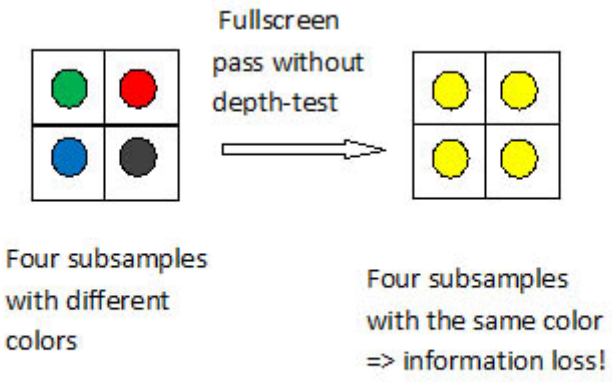

Figure 2

그 이유는 한 픽셀 당 한번씩만 픽셀 셰이더(shader)를 사용할 수 있고, 그 전체 픽셀에 적용되기 때문이다. 사실상 MSAA 버퍼(buffer)들은 비 MSAA 버퍼로 탈바꿈하여 모든 그 이후의 리졸브 작업은 중복이 되는 것이다.

상응하는 픽셀의 모든 부표본에 같은 색깔이 인식되면, MSAA 뎁스 버퍼 (depth buffer)는 그 해당 물체의 윤곽과 일치하지 않는다는 주의해야 한다.

확실히 비 MSAA 형식으로 아 패스들을 렌더링 하여 리졸브 자체를 방지하는 것이 해답일 것이다. 불필요한 리졸브를 방지하는 몇몇 방법을 추천하고자 한다.

1. 비 MSAA 형식의 주요 프레임 버퍼 (스왑 체인) 을 만든다.
2. 주요 지오메트리를 렌더링하는 중간체 렌더 타켓과 재기스를 발생시키는 것을 만들어라.
3. 중간체 MSAA 렌더 타겟의 리졸브를 비 MSAA 표면에 실행하라.
4. 남아있는 패스들을 위해 렌다 타겟 설정을 계속 변화해 주어라. (Figure 3).

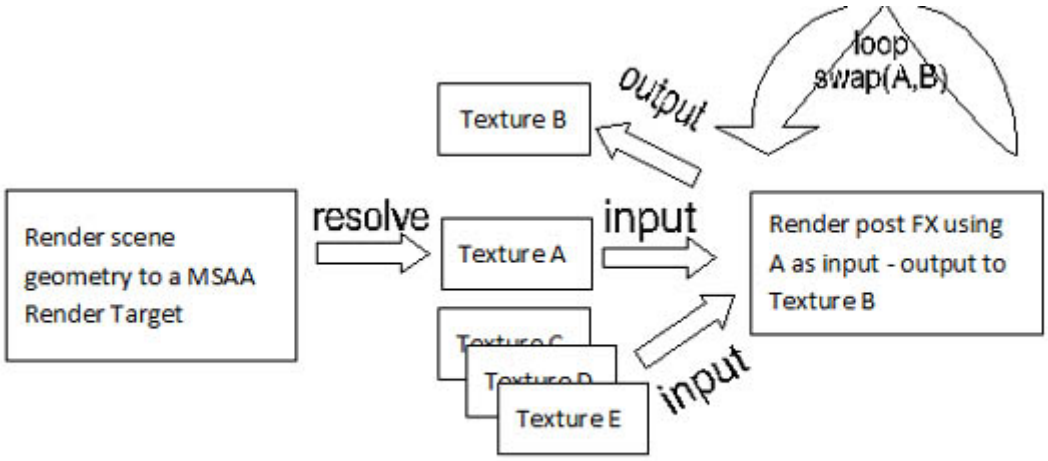

Figure 3

실제의 예를 들어보자. 다음 패스들의 나열은 최근 발매된 한 게임의 분석을 통해 밝혀낸 것이다.

1. M 해당 지오메트리 패스를 주요 렌더 타겟 M 으로 렌더링 한다.
2. M 을 비 MSAA 렌더 차겟 A 로 리졸브한다.
3. A 를 4 분할 최대 화면을 이용하여 M 면에 렌더링 한다.
4. M 을 A 로 리졸브한다.
5. M 에 물 효과로 렌더링 한다.
6. M 을 A 로 리졸브하는 포스트 프로세싱 더 가한다.

한눈에 보더라도 2 단계에서 4 단계까지 중복이 발생한다는 것을 알 수 있다. 특히 3 단계의 경우엔 부표본의 색 정보들이 훼손되는 현상이 발생하기에 화질의 관점에서 보면 사실 권장하고 싶지 않다.

1 단계에서 곧바로 5 단계로 넘어간다면, 2 번의 리졸브 공정을 거치지 않게 된고, 물론 부표본의 색 정보 보존이 가능해 진다.

왜 개발자들은 이러한 점을 발견하지 못한 것일까? 이 질문에 대한 대답은 현대의 엔진들은 객체 지향적이며, 여러 명의 개발자들이 지속적으로 렌더링 코드(rendering code)를 변경했기 때문이다.

사실 어떻게 보면 3 단계만 놓고 보면 사실 나름대로 타당한 포스트 프로세싱 이다. 그렇지만 3 단계를 거침으로 인해 2 단계와 3 단계가 중복됨으로 인해 결국엔 동일한 공정이 되 버리는 것이다. 4 단계에서 리졸브가 이뤄지는 것은 M 이 우연하게 5 단계에서 인풋으로 포함되기 적용되기 때문이다.

보다시피, 중복된 리졸브을 렌더링 파이프 라인에 접목하는 것은 매우 쉽다. 프레임에서 여러 다양한 패스들이 처리되는 것을 파악하고 있는 것은 매우 유용하다. 예기치 않은 반응에 대비해 정기적으로 불필요한 PIX 를 점검하는 것은 역시 유용할 것이다.

유해리졸브

지연 렌더링 기술은 뎁스 (depth), 포지션 (position), 노멀 (normal), 벨로시티 (velocity), 마테리알 ID (material ID) 와 같은 정보를 중간체 렌더 타겟에 저장하는 것이 보통이다. MSAA 형식으로 위의 정보들이 저장된다면, 그 정보들은 나중에 프레임에서 사용되기 전 리졸브가 되어야 할 것이다.

문제는 고정된 리졸브 공정 (fixed function resolve) 을 통해서는 보통 수준의 부표본 밖에 처리해 내지 못한다는 것이다. 따라서 개발자가 의도했던 결과물을 얻기가 힘들어 질것이며, 그래픽상의 왜곡된 효과가 발생할 수도 있다.

마테리얼 ID (material ID) 가 리졸브 되는 경우를 보자. 마테리알 ID (material ID) 들에 대한 평균화 작업은 전혀 실효성이 없으며, 최악의 겨우 불량 ID 를 생성할 가능성도 있다는 사실에 모두가 동의할거라 생각한다.

그럼 우리는 어떻게 이런 종류의 정보들을 다뤄야 할까? 고정된 리졸브 공정 (fixed function resolve) 방식은 전혀 도움이 되지 않는 다는 점을 감안하고 말이다.

DirectX10 에서는 인풋 텍스쳐의 부표본을 인식하는 픽셀 셰이더 적용이 가능하다. 지연 라이팅 기법의 경우엔, 각 부표본에 따라 라이팅 연산(lighting calculation)을 실행하는 것은 가능하며, 그 결과물에 대한 평균화를 할 수 있다. 그로인해 셰이더는 효과적으로 커스텀(custom) 리졸브를 실행하는 것이다.

DirectX10.1 이 적용되는 하드웨어는 뎁스 버퍼 (depth buffer) 의 부표본에 접근을 가능케 함으로서 기능의 범위를 보다 확대시키며, 그로 인해 연관 없는 뎁스 패스 (depth pass) 의 필요성을 사라지게 하는 것이다.

고정된 리졸브의 정보를 사용함으로 인해 문제를 겪는 또 다른 대표적인 경우는 비선형 톤 매핑이다. (색조 매핑/tone mapping) 다중 샘플링 환경에서 톤 매핑을 제대로 실행할 수 있는 한가지 방법은 셰이더를 바탕으로 한 커스텀 리졸브이 적용된 모든 부표본에 톤 매핑을 하는 것이다.

DirectX9 에서는 그게 가능하지가 않다. 그러므로 발생할 수 있는 그래픽 왜곡 현상은 어느 정도 각오를 해야 할지도 모른다. 물론 극상의 뚜렷한 샘플링을 사용하더라도 비슷한 효과를 얻을 수는 있긴 하지만 말이다. 보다 좋은 효과를 얻기 위해서는, 최소한도로 자제를 해야 하긴 하지만, 유해리졸브에 의해 생성된 정보를 기반으로 포스트 프로세싱을 실행할 필요가 있을지도 모른다.

결론

리졸브는 공짜로 이뤄지는 작업이 아니란 것을 아는 것이 중요하다. 리졸브는 명백히 많은 비용이 들어가는 작업이며, 최소한으로 사용을 자제해야 한다. 또한 물론 대부분의 리졸브는 중복되거나 유해하다는 것을 기억할 필요가 있다.

중복되는 것을 피하기 위해선 주요 MSAA 렌더 타겟을 최대한 일찍 리졸브할 필요가 있고, 포스트 프로세싱의 효과를 위해 비 MSAA 형식으로 작업할 필요가 있다는 것을 기억해야 한다. 셰이더를 바탕으로 한 커스텀 리졸브를 적용하여, 양질의 포스트 프로세싱과 지연 렌더링 기술을 제대로 적용하는 것 역시 중요할 것이다.

수많은 렌더링 패스들을 보고 있자면, 무슨 일들이 벌어지고 있는지 파악이 되지 않을 때가 있다. 리졸브를 제대로 이루기 위해서는 규칙적인 점검과 분석이 필수란 사실을 기억하라!