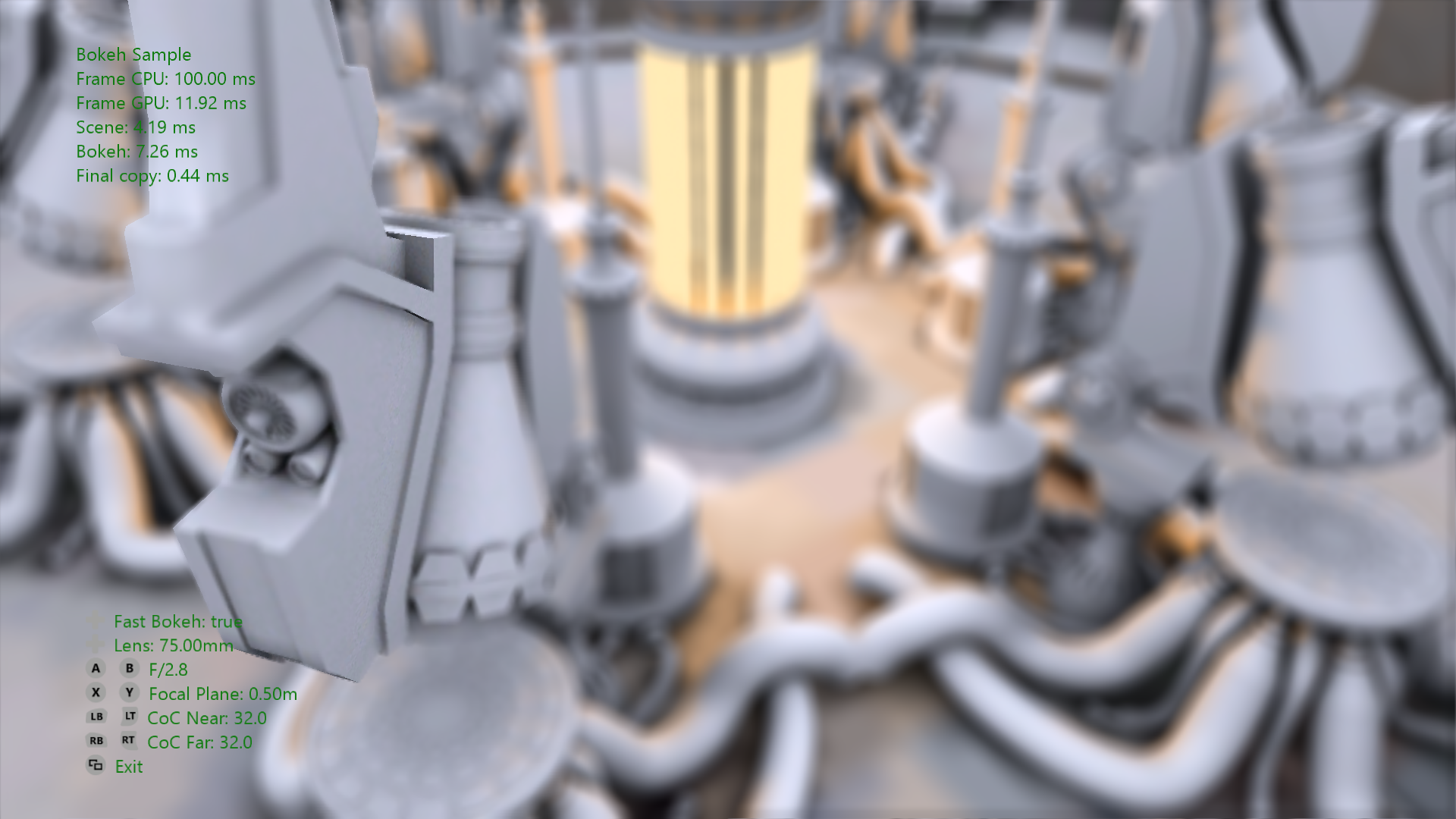


Bokeh 샘플 (DirectX 12)

*\*이 샘플은 2016 년 8 월 Xbox One XDK와 호환됩니다.*

# 설명

이 샘플은 포인트 스프라이트 렌더링을 사용하여 피사계 심도 효과를 만드는 방법을 보여줍니다.



# 샘플 사용하기

이 샘플에서는 다음 컨트롤을 사용합니다.

## 컨트롤

|  |  |
| --- | --- |
| 액션 | 게임 패드 |
| 카메라 회전 | 왼쪽 엄지 스틱 X 축 |
| 카메라 고도 | 왼쪽 엄지 스틱 Y 축 |
| 카메라 거리 | 오른쪽 썸스틱 Y 축 |
| 초점 거리 | D 패드 왼쪽/오른쪽 |
| 초점면 | X 버튼 / Y 버튼 |
| F- 스톱 | A 버튼 / B 버튼 |

|  |  |
| --- | --- |
| 액션 | 게임 패드 |
| 최대 근접 CoC 크기 | 왼쪽 어깨 / 왼쪽 트리거 |
| 최대 Far CoC 크기 | 오른쪽 어깨 / 오른쪽 트리거 |
| Fast Bokeh 셰이더 토글 | D- 패드 업 |
| 사전 설정 샷 순환 | D- 패드 다운 |
| 샘플 종료 | 보기 버튼 |

# 구현 정보

카메라로 찍은 사진에서 중점 밖의 부분을 흐리게 처리하는 것은 이미지의 특정 영역에로 시청자의 주의를 끄는 중요한 효과입니다. 이미지에서 뚜렷한 선명도 영역을 피사계 심도 (DOF)라고 하며 앞면과 뒷면의 영역이 흐리게 나타납니다.

그러한 중요한 예술적 도구가 컴퓨터 그래픽 이미지에서 그 길을 찾았다는 것은 놀라운 일이 아닙니다. 사진에서 DOF는 렌즈 초점 거리, 조리개 및 피사체까지의 거리에 의해 결정되며 대략 얇은 렌즈 방정식에 의해 제공됩니다. 컴퓨터 그래픽에서는 DOF와 같이 임의의 매개 변수와 수식을 사용할 수 있지만 사진과 같이 정의하는 것이 편리합니다.

DOF는 렌즈 및 나머지 이미징 시스템을 통해 볼 때 세계의 한 지점이 카메라의 필름이나 센서에 원으로 투영되기 때문에 존재합니다. 선명한 초점 내 영역에 있는 작은 하나 또는 매우 큰 블러가 있는 영역에서 잘려있기는 하지만 항상 원형입니다. 그 원을 착란원(CoC)이라고 합니다.

일반적으로 DOF의 실시간 그래픽 구현, 픽셀 당 CoC를 계산한 다음 그에 따라 이미지를 흐리게 처리합니다 (오프라인 DOF 구현은 다른 방식으로 작동함). 이러한 실시간 블러는 사실적으로 보이고 유용하기 위해 다음 제한을 만족해야 합니다.

1. 각 점은 CoC 크기의 카메라 조리개의 이미지로 스크린에 투영됩니다. 결과 모양은 일반적으로 Bokeh DOF라고 합니다.

2. 블러링 된 포인트는 맨 뒤에서 앞으로 섞여서 적어도 카메라에서 멀리 떨어진 픽셀과 더 가까운 카메라 픽셀로 인해 흐려진 원으로 인해 중점을 맞춘 이미지가 가려지지 않도록 해야 하며 초점이 맞고 멀리 초점이 맞은 이미지 위에 섞여야 합니다. 이 기능이 없으면 이미지의 초점이 맞은 부분의 가장자리 주위에 색 번짐이 생깁니다.

3. 부분 폐색 - 광학 시스템이 큰 구멍을 통해 장애물을 볼 수 있습니다. 즉, 가까운 필드의 큰 CoC가 아래에 있는 물체를 나타냅니다.

4. 매우 큰 크기의 CoC가 가능해야 하며 고속이어야 합니다.

이 샘플에서 설명된 알고리즘은 3 번을 제외한 위에 나열된 모든 포인트를 충족시킵니다. 부분 폐색을 올바르게 처리하려면 폐색물 뒤에 장면을 렌더링하거나 (깊이 필링과 유사) 또는 폐색된 객체의 색상을 부분 폐색 영역에 묻어야 합니다. 이 샘플에서는 성능상의 이유로 이 단계를 수행하지 않고 대신 기본 핀홀 이미지를 사용하여 일부 장소에서 약간의 시각적 결함을 유발할 수 있습니다.

## 알고리즘:

개념은 간단합니다. 입력의 픽셀을 가져와서 CoC를 계산하고 CoC의 크기이고 소스 픽셀로 채색된 포인트 스프라이트를 출력합니다. 우리는 폐쇄 순서대로 그들을 정렬하고 모든 스프라이트를 모으고 정확하게 흐린 이미지를 얻습니다.

이렇게 간단한 방법으로 구현하면 효과가 환상적으로 보일 수 있지만 현대 하드웨어에서는 실시간으로 만들 수 없으므로 아래는 최적화 된 알고리즘입니다.

1. 원본 색상과 깊이를 RGBZ 텍스처로 변환합니다. 소스 텍스처의 크기는 W\*2 x H\*2 입니다.

2. 텍스처를 한 차례 다운 샘플링합니다. 평균 4가지 색상과 최소 4 가지 깊이를 취합니다.

3. WxH 다운 샘플링 텍스처의 경우 6 개의 뷰포트로 렌더 타겟을 설정하십시오: WxH, W/2xH/2 및 W/4xH/4

4. 렌더 W\*H/4 포인트 프리미티브

1. 각 포인트 프리미티브에 대해 다운 샘플링된 소스 텍스처의 4 개의 RGBZ 값을 읽습니다.

2. 4 픽셀이 많이 다르지 않으면 하나의 스프라이트를 출력하고, 그렇지 않으면 4 개의 스프라이트를 출력합니다

3. 스프라이트의 크기에 따라 해당 뷰포트로 라우트되며, "가까운" 세트의 뷰포트로 들어가는 인포 커스 영역보다 가까운 스프라이트가 더 "멀리" 있는 뷰포트로 이동합니다

5. 6 개의 뷰포트 렌더링을 완료한 후 초점이 맞은 이미지를 재조합하여 초점이 맞은 픽셀이 멀리 흐린 영역을 오버라이드하고 가까운 픽셀이 초점이 맞은 픽셀과 먼 픽셀을 혼합하는지 확인합니다.

가까운 흐려진 픽셀의 혼합을 개선하려면 이미지의 에너지 보존이 중요합니다. 각 소스 픽셀은 단위 에너지를 가지므로 포인트 스프라이트로 스플래팅 한 후에 그 에너지는 스프라이트 영역에 분산됩니다. 조리개 텍스처와 래스터화 규칙이 다르기 때문에 모든 크기와 스프라이트 원점에 대해 닫힌 형태로 가중치를 계산하는 것은 불가능합니다. 샘플은 조리개 스프라이트를 다른 크기로 렌더링하고 결과로 나오는 가중치를 계산한 다음 정규화 인자를 계산합니다.

# 업데이트 기록

2015: 원본 샘플 저작

2018년 11월 : 새 샘플 템플릿을 위해 다시 작성되었습니다.

# 개인정보처리방침

샘플을 컴파일하고 실행할 때 샘플 실행 파일의 이름이 Microsoft로 보내져 샘플 사용을 추적 할 수 있습니다. 이 데이터 수집을 거부하려면 Main.cpp에서 "샘플 사용 텔레메트리"라는 코드 블록을 제거하면 됩니다.

Microsoft의 개인 정보 취급 방침에 대한 일반적인 내용은 [Microsoft 개인 정보 취급 방침](https://privacy.microsoft.com/ko-kr/privacystatement/)을 참조하십시오.