

Meshlet 인스턴스 샘플

*이 샘플은 Microsoft 게임 개발 키트 미리 보기(2019년 11월) 및 PC용 Windows 10 “20H1” 참가자와 호환됩니다.*

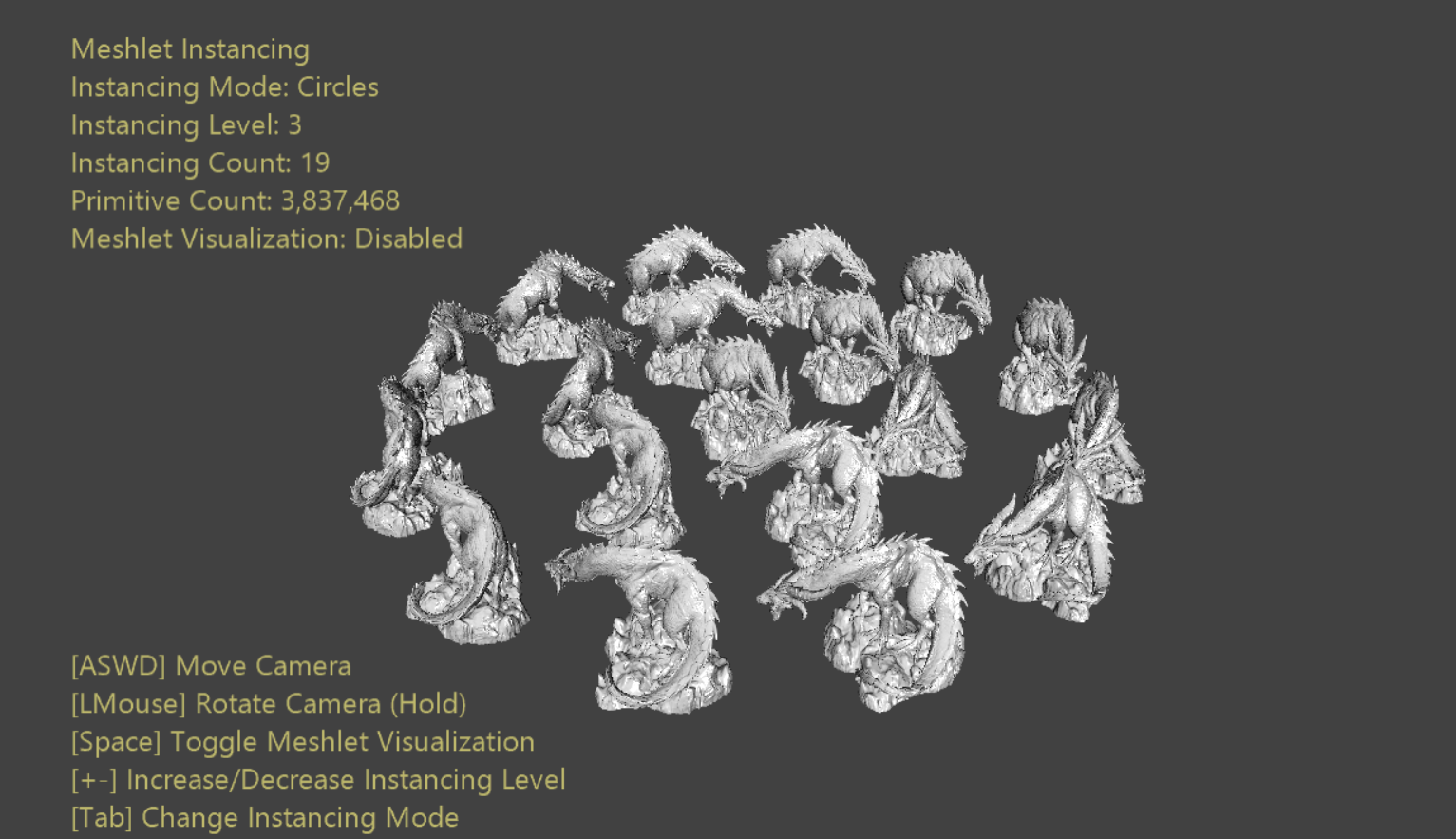
# 설명

메시 셰이더 파이프라인은 인덱스 처리 수행 및 기능 인스턴스화를 담당하던 입력 어셈블러와 비슷합니다. 따라서 메시 셰이더 파이프라인은 기존 그래픽 파이프라인처럼 인스턴스화를 위한 상위 수준의 인터페이스를 제공하지 않으므로 대신 개발자가 메시 셰이더 스레드 ID를 사용하여 고유한 인스턴스화 솔루션을 구현해야 합니다.

GPU는 *웨이브*라는 스레드의 고정 크기 청크로 워크로드를 디스패치합니다. 이 크기는 아키텍처마다 고유하지만 디스패치할 수 있는 스레드의 최소 양에 해당합니다. 입력 어셈블러에서 수행한 기능 중 하나는 작업과 함께 이러한 웨이브를 압축하여 스레드 사용률을 최적화하는 것이었습니다*. 웨이브 사용률*은 디스패치된 총 웨이브 수에 대한 실제 데이터를 처리하는 웨이브 스레드의 비율입니다.

메시 셰이더 파이프라인은 입력 어셈블러 사용을 건너뛰므로 작업과 함께 웨이브를 압축하는 문제는 개발자에게 달려 있습니다. Meshlet은 메시를 친숙한 작업 청크로 사전 처리하여 웨이브 사용률을 최적화하는 도구입니다. 그러나 meshlet이 꽉 차지 못하게 하는 하드 제약 조건은 없으므로 차선책에 해당하는 웨이브 사용률이 유지됩니다. 특히 메시는 가득 차기 전에 기본 형식이 고갈되므로 메시의 마지막 망상 조직이 가득 찰 가능성은 낮습니다. 이러한 점은 메시의 meshlet 수가 0에 가까워질 때 문제로 나타납니다. 이와 같이 잎, 머리카락, 파티클과 같은 작은 메시를 과도하게 인스턴스화하는 경우가 일반적입니다.

이 샘플에서는 채워지지 않은 마지막 meshlet의 여러 인스턴스를 단일 스레드 그룹으로 압축하여 효율적으로 인스턴스화하는 일반적인 방법을 제공합니다. 이 경우 디스패치되는 비활성 스레드의 수가 최소화되어 웨이프 사용률이 100%에 가깝게 유지됩니다.



# 샘플 빌드

Project Scarlett을 사용하는 경우 활성 솔루션 플랫폼을 Gaming.Xbox.Scarlett.x64로 설정하세요.

적절한 하드웨어 및 Windows 10 릴리스가 있는 PC를 사용하는 경우 활성 솔루션 플랫폼을 Gaming.Desktop.x64로 설정합니다.

이 샘플은 Xbox One을 지원하지 않습니다.

*자세한 내용은 GDK 문서에서* 샘플 실행하기*를 참조하세요.*

# 샘플 사용

이 샘플에서는 동심원 및 정육면체의 두 가지 인스턴스화 레이아웃 중에서 선택할 수 있는 옵션을 제공합니다. 이를 통해 다양한 비율로 크기를 조정할 수 있습니다. 인스턴스의 기본 meshlet 구조에 플랫 음영을 지정하는 경우와 시각화하는 경우 간을 토글할 수도 있습니다.

# 컨트롤

|  |  |  |
| --- | --- | --- |
| 작업 | 게임패드 | 키보드 |
| 카메라 이동 | 왼쪽 썸스틱 | WASD 또는 화살표 키 |
| 카메라 회전 | 오른쪽 썸스틱 | LMB + 마우스 누르기 |
| 카메라 초기화 | 오른쪽 썸스틱(푸시) | - |
| 인스턴스화 모드 변경 | A | 탭 |
| Meshlet 시각화 토글 | X | 스페이스바 |
| 인스턴스화 수준 높이기 | 오른쪽 숄더 | + |
| 인스턴스화 수준 낮추기 | 오른쪽 트리거 | - |
| 종료 | 보기 단추 | Esc |

# 구현 참고 사항

메시 셰이더 인스턴스화는 작업을 완료하고 제공된 ID를 사용하여 처리할 올바른 Meshlet 및 인스턴스를 결정하는 데 충분한 셰이더 인스턴스를 디스패치하는 간단한 작업입니다. 선택한 인덱싱 스키마는 어떤 스레드 그룹이 어떤 작업을 처리할 것인지 결정합니다.

스레드 그룹당 하나의 Meshlet을 가정할 때 가장 간단한 구현은 MeshletCount \* InstanceCount개의 스레드 그룹을 디스패치하는 것입니다. 이 개수는 워크로드를 완전하게 처리하는 데 충분한 스레드 그룹이 될 수 있습니다. 또한 다음과 같은 아주 간단한 인덱스 스키마를 제공합니다.

* *MeshletIndex* = *GroupID.x* / *InstanceCount*
* *InstanceIndex* = *GroupID.x* & *InstanceCount*

즉, 첫 번째 *InstanceCount*개의 스레드 그룹은 Meshlet *0*을 모두 처리하지만 다른 인스턴스를 처리합니다. 좀 더 중요한 점은 마지막 *InstanceCount*개의 스레드 그룹이 Meshlet*(MeshletCount – 1)*을 모두 처리하게 된다는 것입니다. 첫 번째*(MeshletCount-1)* Meshlet은 '가득 채워지고'(최대에 가까운 정점 및/또는 기본 형식 포함) 마지막 Meshlet은 덜 채워진다고 가정합니다. 따라서 마지막*(MeshletCount – 1)* 스레드 그룹은 첫 번째*(MeshletCount – 1) \* InstanceCount*개의 스레드 그룹보다 덜 최적화된 웨이브 사용률을 유지합니다. 충분히 스파스 상태인 여러 인스턴스를 단일 스레드 그룹으로 압축할 수 있습니다.

실제로 스레드 그룹 크기가 최대 Meshlet 크기와 같은 경우에는 개의 인스턴스를 단일 스레드 그룹에 맞출 수 있습니다. 따라서 마지막 Meshlet을 처리하려면 압축 스레드 그룹이 필요합니다. 이를 통해 셰이더의 이러한 스레드 그룹에 대해 약간의 ALU 비용을 추가로 들여 최상의 웨이브 효율성을 얻을 수 있습니다.

# 업데이트 기록

2019년 10월 31일 – 샘플 만들기

# 개인정보처리방침

샘플을 컴파일하고 실행할 때 샘플의 사용을 추적하는 데 도움이 되도록 샘플 실행 파일의 파일 이름이 Microsoft에 전송됩니다. 이 데이터 수집을 옵트아웃하려면 Main.cpp에서 "샘플 사용 원격 분석"이라고 레이블이 지정된 코드 블록을 제거할 수 있습니다.

Microsoft의 일반 개인정보취급방침에 대한 자세한 내용은 [Microsoft 개인정보처리방침](https://privacy.microsoft.com/en-us/privacystatement/)을 참조하세요.