**# 텍스쳐(Texture)**

**텍스쳐링, 텍스쳐매핑**

**=> 표면을 모델링 하지 않고 표면에 이미지(텍스쳐)를 그리는 방법**

**\*픽셀 => 렌더타겟에 그려질 하나의 점**

**\*텍셀 => 텍스쳐의 픽셀들의 색깔 정보(Texture + Pixel)**

**텍스쳐 = 텍셀을 저장하기 위한 구조화된 데이터의 집합**

**텍스쳐 매핑을 위해서는 샘플러가 필요하다.**

**텍스쳐는 배열로도 사용이 가능하고 다중 샘플 텍스쳐도 가능**

**# 밉맵**

**해상도가 레벨별로 절반이 되는 순서화된 일련의 텍스쳐들**

**# 텍스쳐 좌표**

**원하는 것들을 통과하거나 제거 => 텍스쳐 필터링**

**모든 텍스쳐 이미지에 대해서 텍셀 좌표계를 무조건 통일된 것으로 쓰자.. 0 ~ 1.f**

**UV좌표계**

**3D는 u, v, w**

**UV는 모델좌표계도 월드좌표계도 아니다. 텍스쳐 좌표계이다. => 정점의 변동을 그대로 반영한다.**

**텍스쳐 좌표도 픽셀의 Interpolation이 일어난다.**

**#**

**보간을 위해서**

1. **밉맵 레밸 결정**
2. **텍셀 결정**

**# 샘플링과 필터링**

**샘플러라는 객체가 필요하다.**

**(UV좌표 -> 픽셀좌표 해줌)**

**샘플링 => 텍셀 대상 선택하는 작업**

**필터링 => 텍셀 정보를 바탕으로 색을 만들어내는 작업**

**# 텍스쳐 압축**

**장점 : 적은 VRAM으로도 동일한 퍼포먼스를 보여줄 수 있다.**

**단점 : 압축된 텍스쳐를 푸는 시간으로 인한 지연이 있을 수 있다.**

**-텍스쳐 압축 보간**

**블록 압축(Block Compression )=> DirectX에서 사용하는 방법.**

**하지만, 가로 세로가 무조건 4의 배수가 되어야 한다.**

**Locality => 주변에 있는 메모리**

**텍스쳐 매핑 8시 정도 연산을 줄일 수 있다. –**

**#앞축은 따로 쓰지 않겠다.**

**BC => Block Compression**

**DXGI\_FORMAT\_BC1\_**

**RGB 비트 비율 5:6:5 , 0 or 1 알파**

**DXGI\_FORMAT\_BC2 ~\_BC3**

**Alpha값이 추가된다. (4비트 ~ 8비트)**

**# DDS (Direct Draw Surface)**

**압축 기능 뿐만 아니라 밉맵도 제공한다.**

**# 서브리소스(Subresource)**

* **텍스쳐 리소스의 부분집합**

**리소스 => 그림을 그리기 위한 데이터.**

**대표적으로 텍스쳐(밉맵의 집합)**

**다렉은 리소스를 텍스쳐를 하나의 텍스쳐 뿐만 아니라 텍스쳐들(서브리소스)의 배열로 취급**

**배열 슬라이스, 밉 슬라이스**

**서브리소스 인덱스 = MipSlice + (ArraySlice \* MipLevels)**

**인덱스 값을 구하는 함수를 제공함 => D3D12CalcSubrecource**

* **버퍼에는 서브리소스가 하나만 존재한다**

**.**