**DirectXTex 텍스처 처리 라이브러리**

<https://github.com/microsoft/DirectXTex/wiki>

이 라이브러리는 여러 가지의 텍스쳐 파일을 읽거나 쓰기 위한 공유 소스 라이브러리로 텍스처 크기 조정, 형식 변환, 밉맵 생성, Direct 3D 런타임 텍스처 리소스에 대한 블록 압축, Height 맵과 Normal맵으로 변환 등 다양한 텍스처 콘텐츠 처리 작업을 수행합니다.

지원 파일 형식으로는 .DDS파일, .TGA파일,. HDR파일을 지원하고 WIC(Windows Image Component) API를 사용한다

|  |
| --- |
| WIC란?  C, C++에서 이미지 관련 기능을 제공합니다. 표준 메타 데이터 형식에 대한 기본 제공 지원(BMP, .JPEG, .PNG, TIFF, GIF 등등) 기본적인 이미지 파일에 대한 메타데이터와 이미지 데이터 정보를 얻거나 파일을 생성할 수 있다. 수 있는 API라고 할 수 있다. |

이렇기 때문에 윈도우에서 사용되는 대부분의 이미지 파일의 메타데이터 추출과 이미지 데이터 로드 하고 이것을 수정하여서 메모리와 디스크에 파일을 생성하는 기능을 제공하고 있다.

이렇게 여러 확장자들에 대한 공통 함수로는

|  |
| --- |
| GetMetadataFrom(확장자)Memory - 메모리의 (확장자) 지원 파일에서 메타데이터를 추출합니다.  GetMetadataFrom(확장자)File - 디스크의 (확장자) 지원 파일에서 메타데이터를 추출합니다.  LoadFrom(확장자)Memory - 메모리의 (확장자) 지원 파일에서 이미지 데이터를 로드합니다.  LoadFrom(확장자)File - 디스크의 (확장자) 지원 파일에서 이미지 데이터를 로드합니다.  SaveTo(확장자)Memory - 메모리에 (확장자) 지원 파일을 생성합니다.  SaveTo(확장자)File - 디스크에 (확장자) 지원 파일을 생성합니다. |

이렇게 존재한다 보다시피 파일의 정보를 가져오거나 정보를 토대로 파일을 저장하고 수정하는 기능을 하는 라이브러리라는 것을 알 수 있다.

그래서 이번에는 추출과 저장을 어떤 함수를 통해서 하는지 알아냈으니 추출한 이미지 데이터와 메타데이터를 사용해서 텍스처를 변환하는 함수에 대하여 조사를 해봤다.

|  |
| --- |
| FlipRotate – 이미지 또는 다수의 이미지를 반전하거나 회전시킨다.  Resize – 이미지의 크기를 조정한다  Convert – 한 픽셀 형식에서 다른 픽셀의 형식으로 변환한다.  ConvertToSinglePlane – 평면 이미지를 단일 평면 픽셀 형식으로 변환한다.  GenerateMipMaps – 이미지에 대한 밉맵 생성  GenerateMipMaps3D – 슬라이스를 표현하는 이미지들에 3D볼륨 텍스처에 대한 밉맵을 생성한다.  ScaleMipMapsAlphaForCoverage – 알파 적용 범위를 유지하기 이해 밉맵의 크기를 조정합니다.  PremultiplyAlpha – PremultiplyAlpha를 사용하여서 변환  Compress – 이미지를 BC(block Compressed)형식으로 압축한다.  Decompress – 이미지를 BC 형식이 아닌 이미지로 압축 해제  ComputeNormalMap – Height맵을 Normal맵으로 변환  CopyRectangle – 소스 이미지에서 대상의 이미지로 사각형을 복사합니다. (압축 지원 안함)  ComputeMSE – 두개의 입력 이미지를 기반으로 각 구성 요소에 대한 평균 제곱 오차를 계산한다.  EvaluateImage – 입력 이미지에 대한 사용자 정의 함수를 평가합니다.  TransformImage – 사용자 정의 함수를 사용하여 기존 이미지에 새 이미지 만들기 |

|  |  |
| --- | --- |
| **PremultiplyAlpha**란?  기존의 RGBA방식을 **Straight alpha**라고 부른다. rgba(1, 0.78, 0, 0.4)을 해석하면  R: 100% 강도의 붉은 빛을 내며,  G: 78% 강도의 초록 빛을 내며,  B: 0% 강도의 푸른 빛을 내며  A: RGB 세 가지 각각의 빛의 세기를 그 40% 만큼으로 줄여서, 불투명도를 40%로 떨어뜨린다. (60% 투명해 보이게 한다.)  이렇게 해석을 할 수 있다.  그러나 이번에 처음 등장한 **PremultiplyAlpha**라는 개념을 사용하면 위와 같은 연산을 하지 않고 그래픽 처리의 연산을 줄어드는 장점이 있다. RGB의 각각의 값이 알파의 값과 비교해서 항상 더 작거나 같다는 점이다.  rgba(1, 0.78, 0, 0.4) 이전과 동일한 RGBA를 **PremultiplyAlpha** 방식으로 표현하면 rgba(0.4, 0.31, 0, 0.4)으로 표현할 수 있다.  이를 해석하면 아래로 설명할 수 있다.  R: 40% 강도의 붉은 빛을 내며, (100% 강도의 붉은 빛의 세기를 그 40% 만큼으로 줄여서)  G: 31% 강도의 초록 빛을 내며, (78% 강도의 초록 빛의 세기를 그 40% 만큼으로 줄여서)  B: 0% 강도의 푸른 빛을 낸다. (0% 강도의 푸른 빛의 세기를 그 40% 만큼으로 줄여서)  A: 미리 이 0.4 만큼의 알파 값을 곱해 두었다.  Ref link(https://nanite.tistory.com/98) | |
| Normal Map(Bump Map의 일종)(법선 맵핑)이란?  튀어나온 곳과 움푹 들어간 곳의 빛을 왜곡시키는 기법으로 폴리곤을 그대로 두고 세세한 부분을 추가하기 위해 사용하는 텍스처다.  이렇게 표현하기 위해서 질감의 노말 값의 xyz 값을 rgb로 저장하여 정보를 담고 있는 것이 Normal맵이다. 그래서 Normal 맵의 색이 다양하다. 그리고 이 데이터를 이용해서 빛의 반사각을 조절하여서 질감을 표현한다.  쉽게 이야기해서 low폴리곤의 모델링에 실제로 폴리곤을 늘리지 않고 High폴리곤 모델링처럼 보이게 만드는 기술이다.  노멀 맵  제작 순서는 (그래픽팀에서 제작해줌 아니면 런 해야함)   1. low폴리 모델링 제작 2. High폴리 모델링 제작 후 Normal 맵 생성 3. Normal맵에 low폴리곤 모델을 적용 4. 결과로 low폴리곤 데이터가 high폴리곤 셰이딩으로 보여짐   이렇게 된다. 즉 빛을 pre vertex가 아닌 pre pixel로 라이팅을 하여 픽셀마다 Normal맵 데이터로 그려 넣으면 굴곡 있는 표면을 표현할 수 있다.  그리고 찾다 보니 Bump Map만들어주는 사이트를 발견했다. 나중에 포트폴리오를 만들 때 굉장히 유용해 보였다.    Ref link(https://cpetry.github.io/NormalMap-Online/) |

|  |
| --- |
| Height Map(Bump Map의 일종)(변위 맵핑)이란?  Normal 맵과 다르게 높이차이만 저장을 하고 있기 때문에 그레이 스케일을 가지고 있다.  그래서 Normal 맵과 함께 사용한다.  Height맵은 실제 정점의 위치를 변경하는데 사용이 되고 이런 종류의 변위는 지형 텍스처에 자주 생성되는 방식이다.  변위 맵  그 외의 용도로 **parallax Mapping**이라는 카메라에서 텍스처 좌표의 변위차를 계산하려는 상황에서 계산비용이 많이 들지만 보기 좋은 결과를 내는 매핑이 가능하다.  Ref link(https://cgcookie.com/articles/normal-vs-displacement-mapping-why-games-use-normals) |

먼저 실제로 사용하는 Load함수를 분석해보았다.

먼저 TexMetadata라는 지역 변수를 만들어서 사용하는데

텍스트이(가) 표시된 사진

자동 생성된 설명

라는 구조체이다. 그리고 위에서 찾은 확장자에 맞게 메타데이터를 추출하는 코드를 추가하여서 문제 텍스처의 정보를 집어넣었다. GetMetadataFromWICFile() 함수를 사용하여서 이미지의 정보를 가져왔고 가져온 정보로 desc를 만들고 현재 저장 되어있는 텍스처와 동일한 것이 있는지 확인하고 있을 경우 저장 되어있던 텍스처를 가져온다.

없을 경우는 LoadFromWICFile() 함수를 사용하여서 이미지 데이터를 가져오고 가져온 정보를 토대로 SRV를 만들고 만드는데 여기서 기존에 사용한 CreateTexture함수의 멤버 변수가 기존의 알던 함수와 조금 다른 것을 확인했다.

라이브러리에서 기존에 사용하던 함수의 오버로딩함수를 제공해줬다.

|  |
| --- |
| 라이브러리Helper func  IsSupportedTexture - 주어진 텍스처 메타데이터가 장치에서 지원되면 true를 반환합니다.  CreateTexture - 이미지에서 Direct3D 리소스를 만듭니다.  CreateShaderResourceView – 이미지에서 Direct3D 11 리소스 및 SRV를 만듭니다.  CaptureTexture - Direct3D 렌더 대상을 캡처하고 이미지를 반환합니다. |

이런 추가적으로 오버 로딩된 함수를 사용해서 SRV를 만들고 만들 때 사용했던 desc를 저장하는 것으로 Load 함수가 종료가 된다.