DirectStorage프로그래밍 소개

유영천

https://megayuchi.com

Tw: @dgtman

미리 공지

- DirectStorage를 사용한 프로그래밍 요령을 소개합니다.
- DirectStorage의 모든 것을 다루지 않습니다(못합니다).
- 상세한 공식 문서는 현재 GDKX(for XBOX)로만 제공되기 때문에 id@xbox프로그램에 가입되어 있지 않으면 전적으로 샘플코드에 의존해야합니다.
- DirectStorage는 DirectX 12시스템의 일부입니다. 따라서 DirectX 12프로그래밍을 전혀 모른다면 DirectStorage프로그래밍을 학습할 수도 적용할 수도 없습니다.

DirectStorage?

DirectStorage

- GPU측 메모리에 데이터를 올리려면 기존에는 'storage-> upload buffer->GPU 메모리'의 단계를 거쳐야 했다. (https://youtu.be/lHlAloRa-HI)
- Direct Storage를 사용하면 코드상으로는 'storage -> GPU 메모리'로 직접 올리는 것처럼 작동한다.
- 현재는 PC의 경우 내부적으로 storage->upload buffer->GPU메모리로 전송한다.
- DirectStorage는 NVME storage를 억세스 하는 최적화된 API이다. Std File I/O보다 훨씬 효율적이라고 한다.
- I/O는 비동기/병렬적으로 처리된다.
- Storage에서 읽어서 메모리에 올릴 때 GPU기능을 이용해서 압축해제(일부 압축방식만 지원)하거나 CPU를 이용한 자체 포맷의 압축을 해제할 수 있다.

DirectStorage

- ID3D12Resource로 표현되는 모든 리소스에 대해 적용 가능. Texture, Vertex Buffer, Index Buffer등. 실질적으로는 Texture에 가장 많이 사용될 것으로 보인다.

Architecture

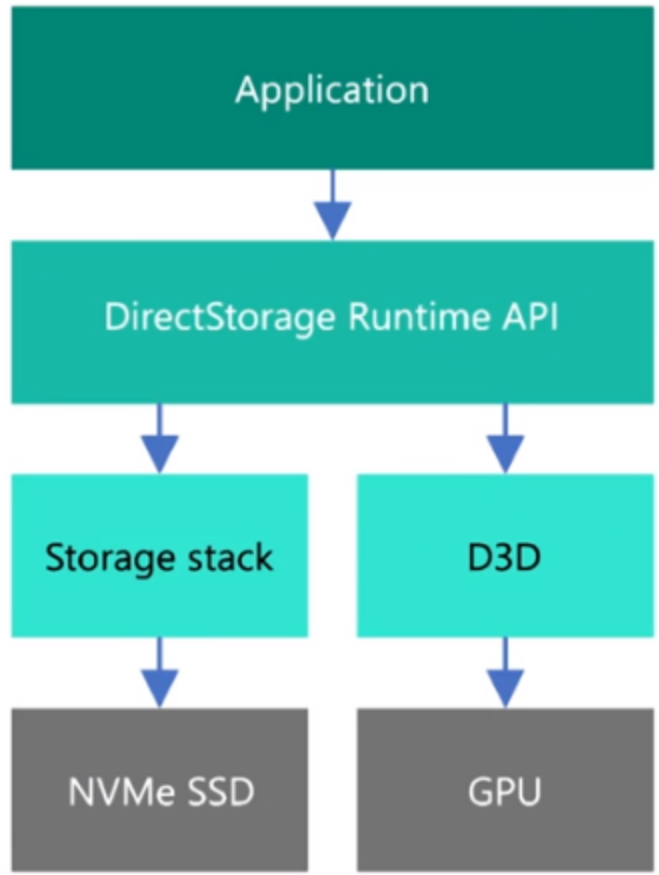
Goals and design principles same as Xbox

- API very similar with only minor differences
- Implementation details differ

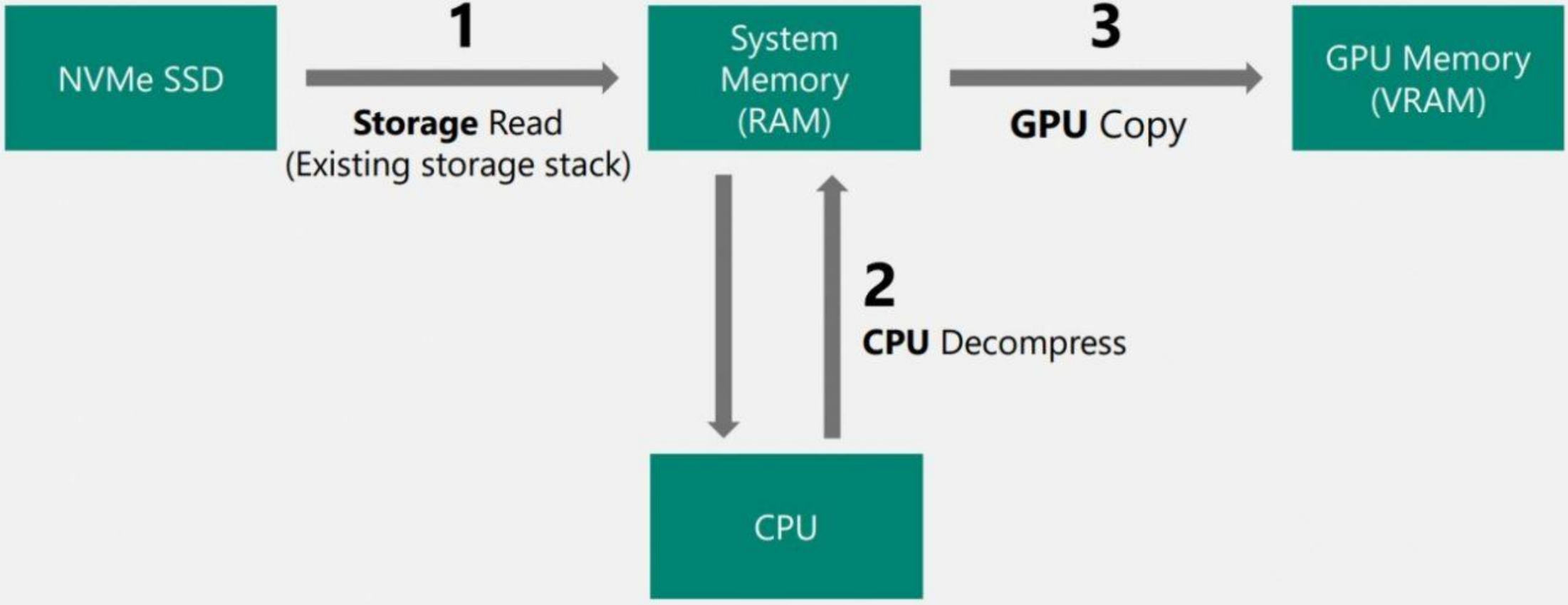
Three primary components

1. DirectStorage Runtime + API
2. Asset decompression
   a) Powered by D3D
3. Windows storage stack

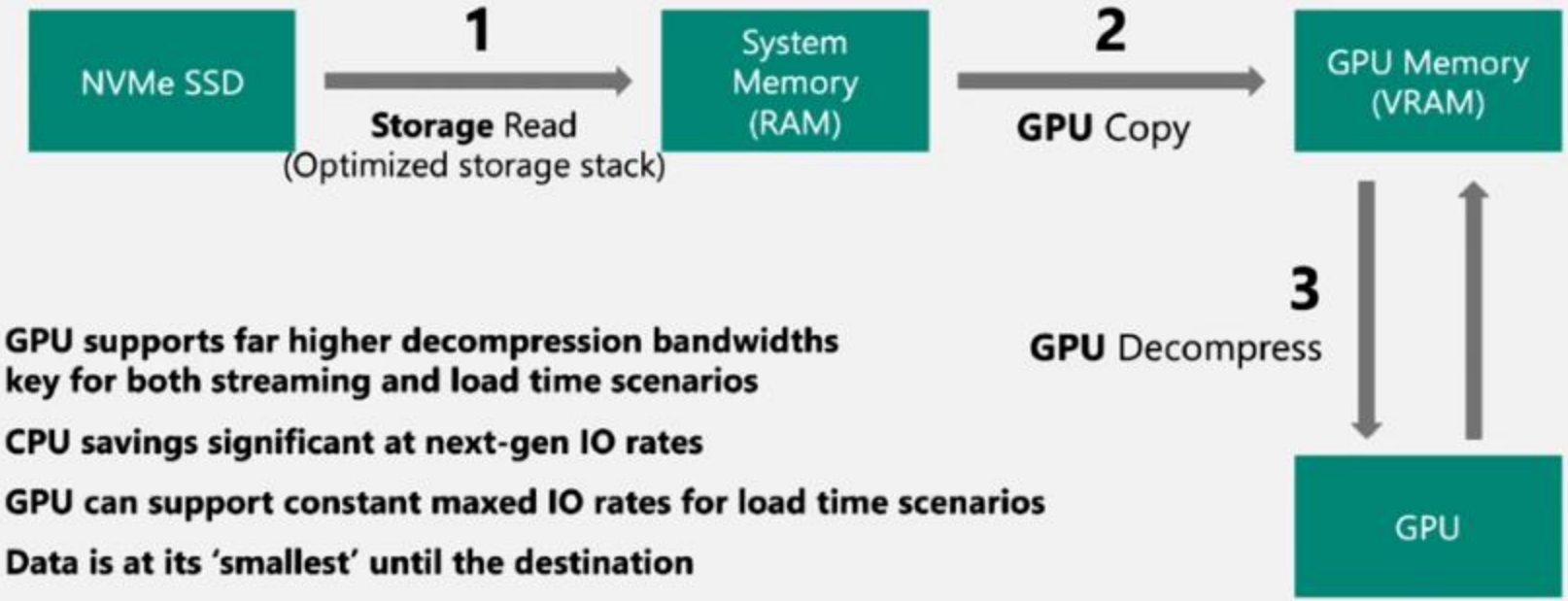
These components work together to optimize end to end data flow



Flow of GPU assets (today)



Flow of GPU assets (with DirectStorage for Windows)



- **GPU supports far higher decompression bandwidths key for both streaming and load time scenarios**
- **CPU savings significant at next-gen IO rates**
- **GPU can support constant maxed IO rates for load time scenarios**
- **Data is at its 'smallest' until the destination**

샘플코드 분석

- microsoft/DirectStorage: DirectStorage for Windows is an API that allows game developers to unlock the full potential of high speed NVMe drives for loading game assets. (github.com)
- DirectStorage/Samples/HelloDirectStorage/
  - 기본적인 작동방법
- DirectStorage/Samples/MiniEngine/ModelViewer/
  - 패키징 파일 만들기
  - 로딩시 압축풀기
  - Storage->D3D12Resource로 업로드

실전적용

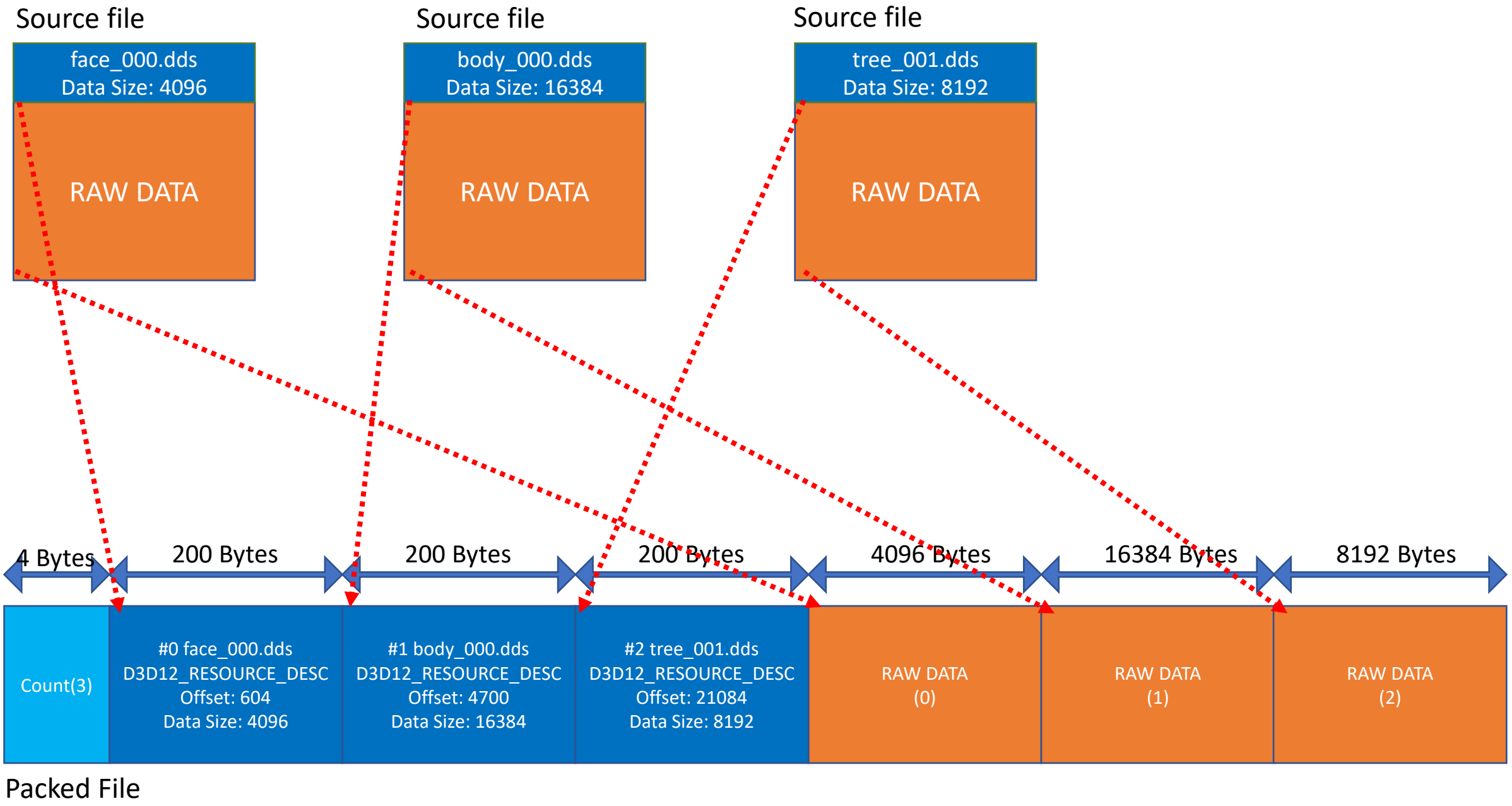
실전적용을 해야 내 기술이 된다.

환경세팅

- NuGet package로 제공.
- Solution Explorer->Manage NuGet Packages for solution -> Browse-> DirectStorage로 검색

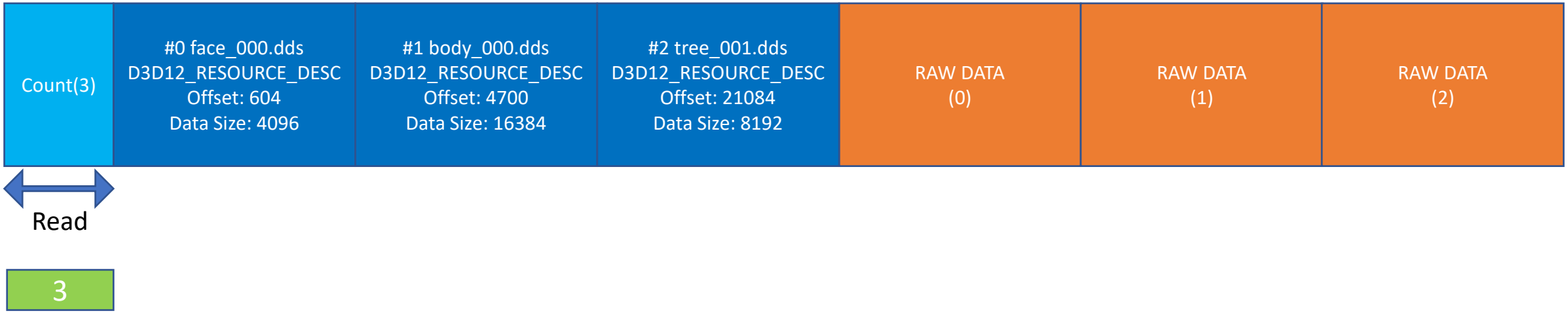
파일 패키징

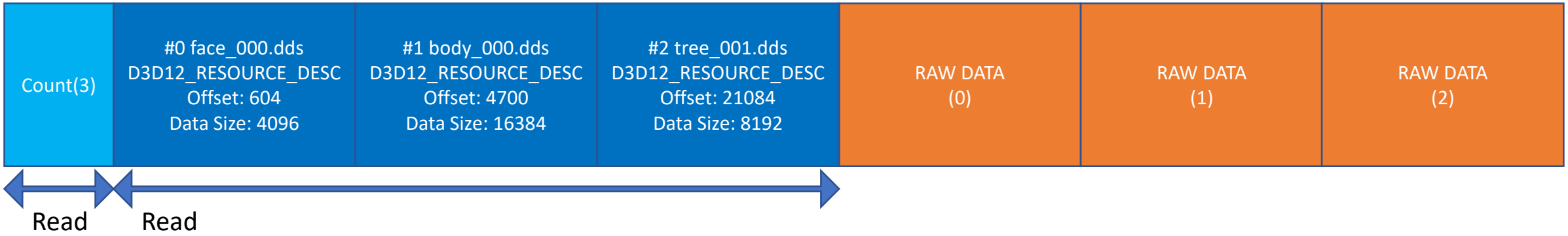
- 개별 파일을 하나하나 로드 하면 DirectStorage를 사용할 이유가 없다.
- 각 파일에는 헤더가 붙어있고 sub resource 사이의 padding이 존재할 수 있기 때문에 원시 파일로는 배치처리를 할 수 없다.
- 결국 데이터를 가공해야 한다.
- 여러 개의 원시 파일의 헤더파트와 raw데이터 파트를 떼서 패키징 파일로 만든다.
- 패키징 파일을 따로 만들고 offset으로 읽어 들이는 시점에서 DirectStorage를 사용하지 않아도 성능이 향상된다.

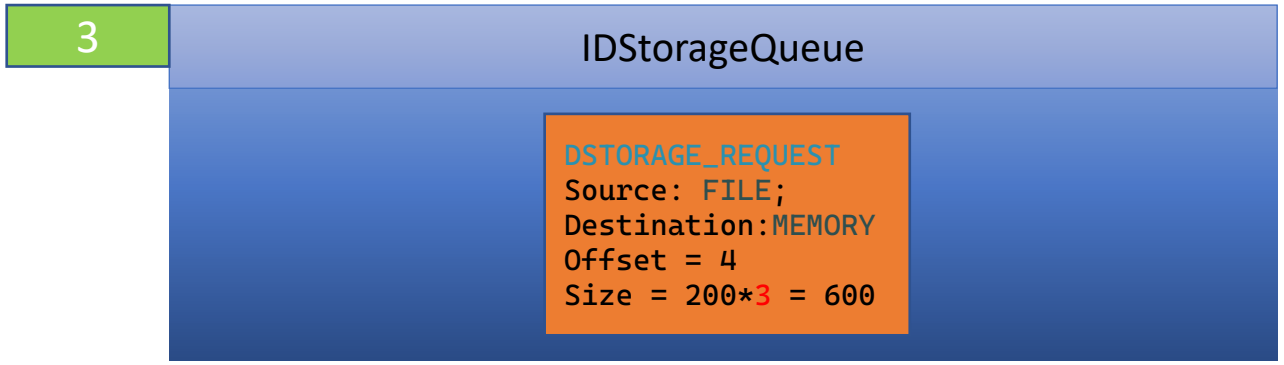

Source file
face_000.dds
Data Size: 4096
RAW DATA
Source file
body_000.dds
Data Size: 16384
RAW DATA
Source file
tree_001.dds
Data Size: 8192
RAW DATA
4 Bytes
200 Bytes
200 Bytes
200 Bytes
4096 Bytes
16384 Bytes
8192 Bytes
Count(3)
#0 face_000.dds
D3D12_RESOURCE_DESC
Offset: 604
Data Size: 4096
#1 body_000.dds
D3D12_RESOURCE_DESC
Offset: 4700
Data Size: 16384
#2 tree_001.dds
D3D12_RESOURCE_DESC
Offset: 21084
Data Size: 8192
RAW DATA
(0)
RAW DATA
(1)
RAW DATA
(2)
Packed File

DirectStorage를 이용한 File -> GPU 로딩

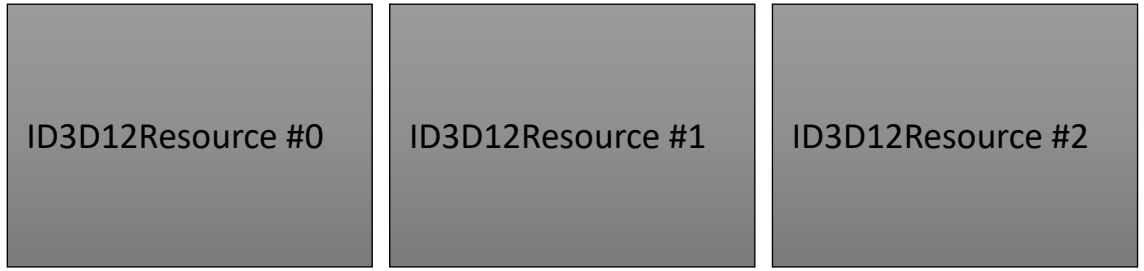
1. 패키징 파일이 몇 개의 데이터를 포함하고 있는지 개수를 얻는다.
2. 1에서 얻은 개수로 각 데이터의 헤더 파트의 메모리를 할당
3. 파일의 데이터(header data)-> 시스템 메모리의 `DSTORAGE_REQUEST`를 `IDStorageQueue`에 enqueue한다.
4. DirectStorage를 이용해서 File -> Memory로 헤더 파트를 batch 로딩.
5. 각 파일의 헤더 데이터를 참조하여 로딩의 목적지가 될 ID3D12Resource를 생성.
6. 파일의 데이터(raw data)-> GPU리소스의 `DSTORAGE_REQUEST`를 `IDStorageQueue`에 enqueue한다.
7. Submit하고 결과를 wait.

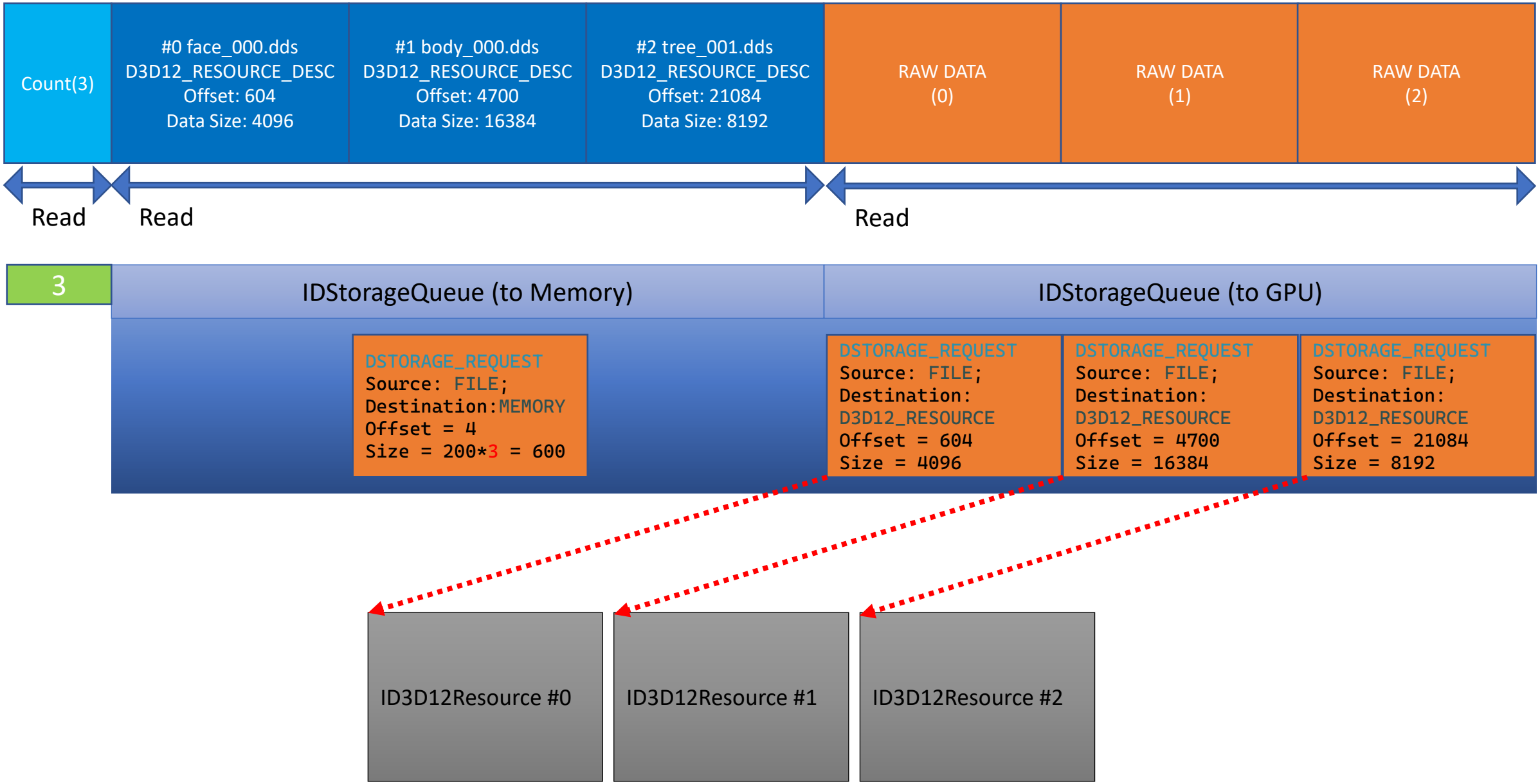

Count(3)
#0 face_000.dds
D3D12_RESOURCE_DESC
Offset: 604
Data Size: 4096
#1 body_000.dds
D3D12_RESOURCE_DESC
Offset: 4700
Data Size: 16384
#2 tree_001.dds
D3D12_RESOURCE_DESC
Offset: 21084
Data Size: 8192
RAW DATA
(0)
RAW DATA
(1)
RAW DATA
(2)
Read
3





Create D3DResources

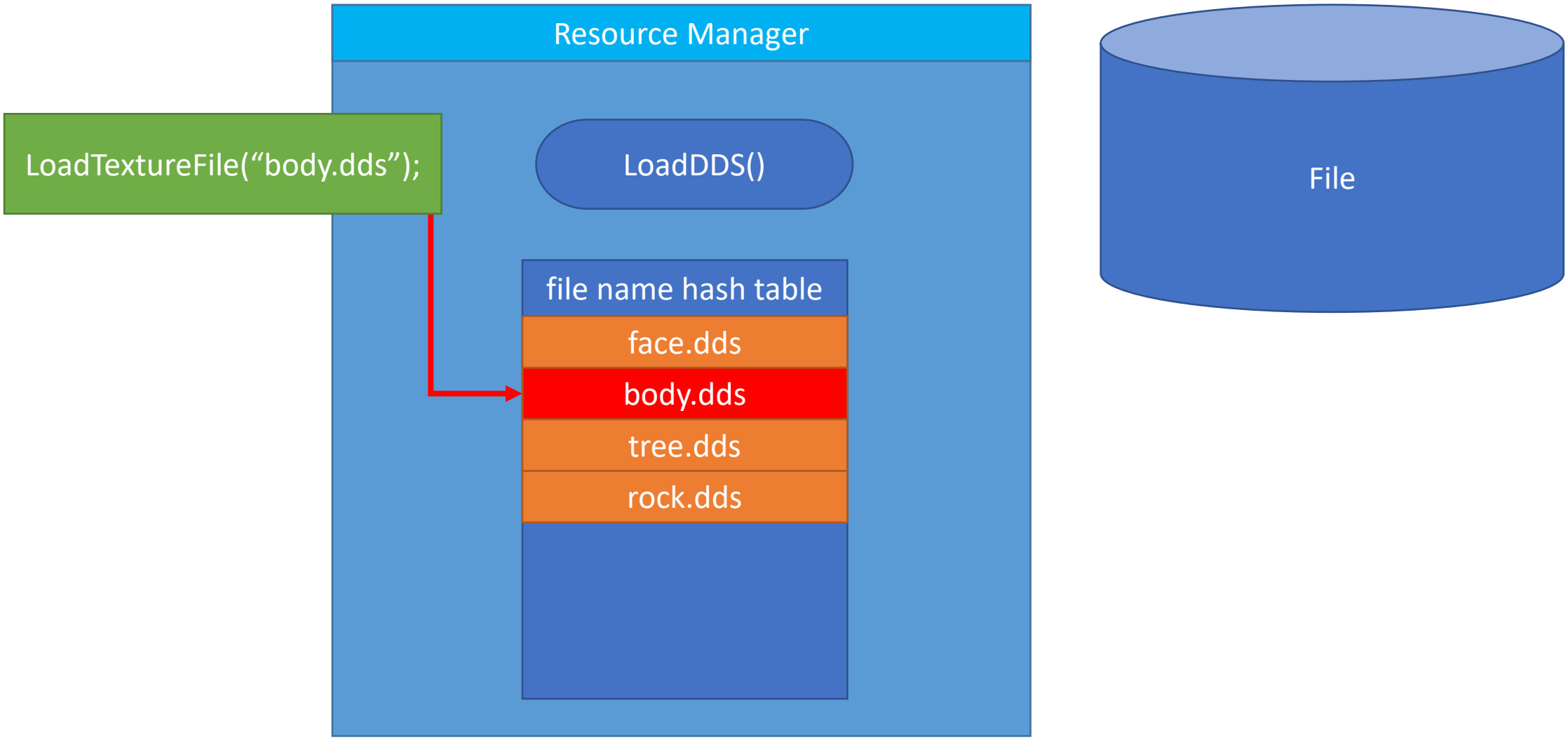



Count(3)
#0 face_000.dds
D3D12_RESOURCE_DESC
Offset: 604
Data Size: 4096
#1 body_000.dds
D3D12_RESOURCE_DESC
Offset: 4700
Data Size: 16384
#2 tree_001.dds
D3D12_RESOURCE_DESC
Offset: 21084
Data Size: 8192
RAW DATA
(0)
RAW DATA
(1)
RAW DATA
(2)
Read
Read
Read
3
IDStorageQueue (to Memory)
IDStorageQueue (to GPU)
DSTORAGE_REQUEST
Source: FILE;
Destination:MEMORY
Offset = 4
Size = 200*3 = 600
DSTORAGE_REQUEST
Source: FILE;
Destination:
D3D12_RESOURCE
Offset = 604
Size = 4096
DSTORAGE_REQUEST
Source: FILE;
Destination:
D3D12_RESOURCE
Offset = 4700
Size = 16384
DSTORAGE_REQUEST
Source: FILE;
Destination:
D3D12_RESOURCE
Offset = 21084
Size = 8192
ID3D12Resource #0
ID3D12Resource #1
ID3D12Resource #2

기존 엔진(자체 엔진)에 적용

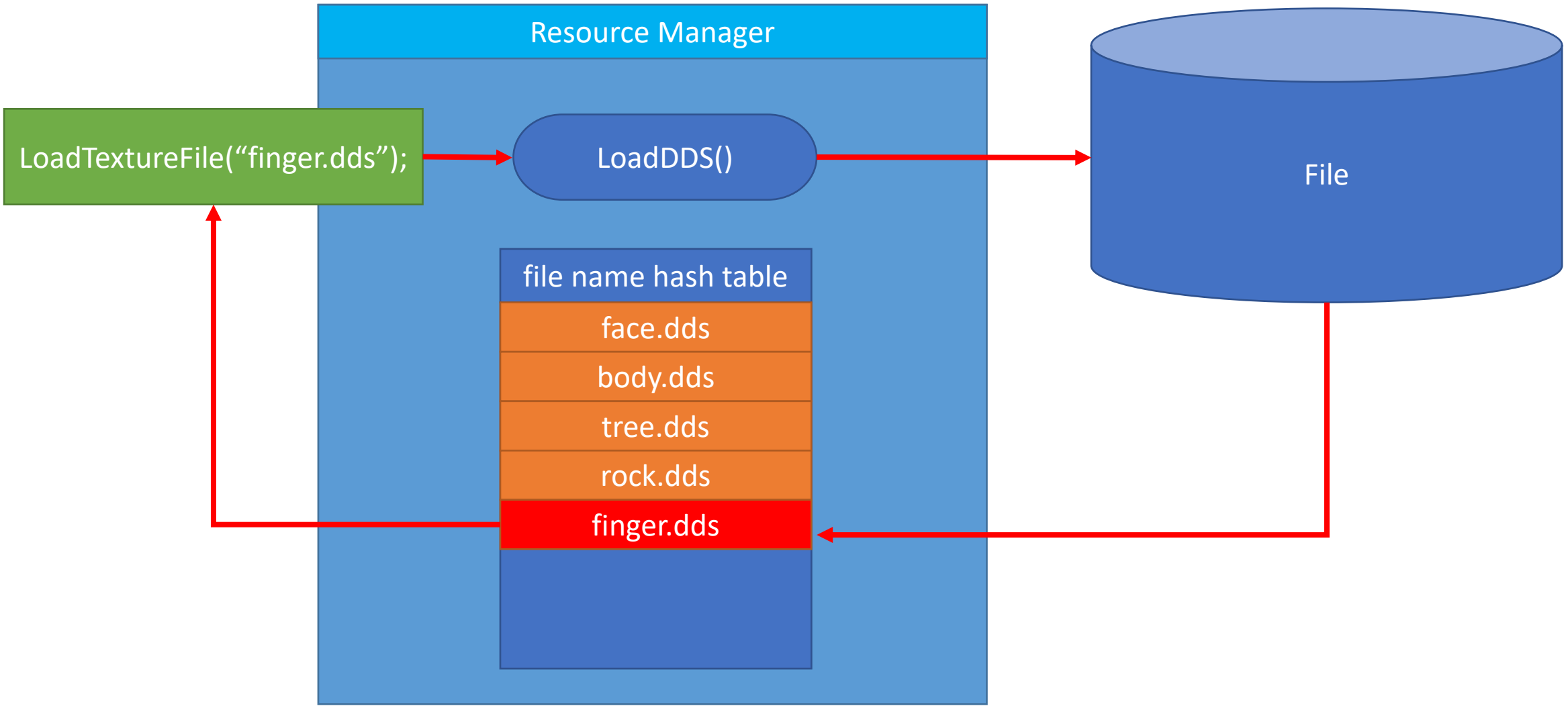
- 텍스처 캐시로 적용
- 기존 텍스처 로딩 함수를 대체

Texture Manager



먼저 Hash table에서 찾아보고 해당 텍스처가 로드되어 있으면 ref count를 증가시키고 텍스처 데이터를 리턴

Texture Manager

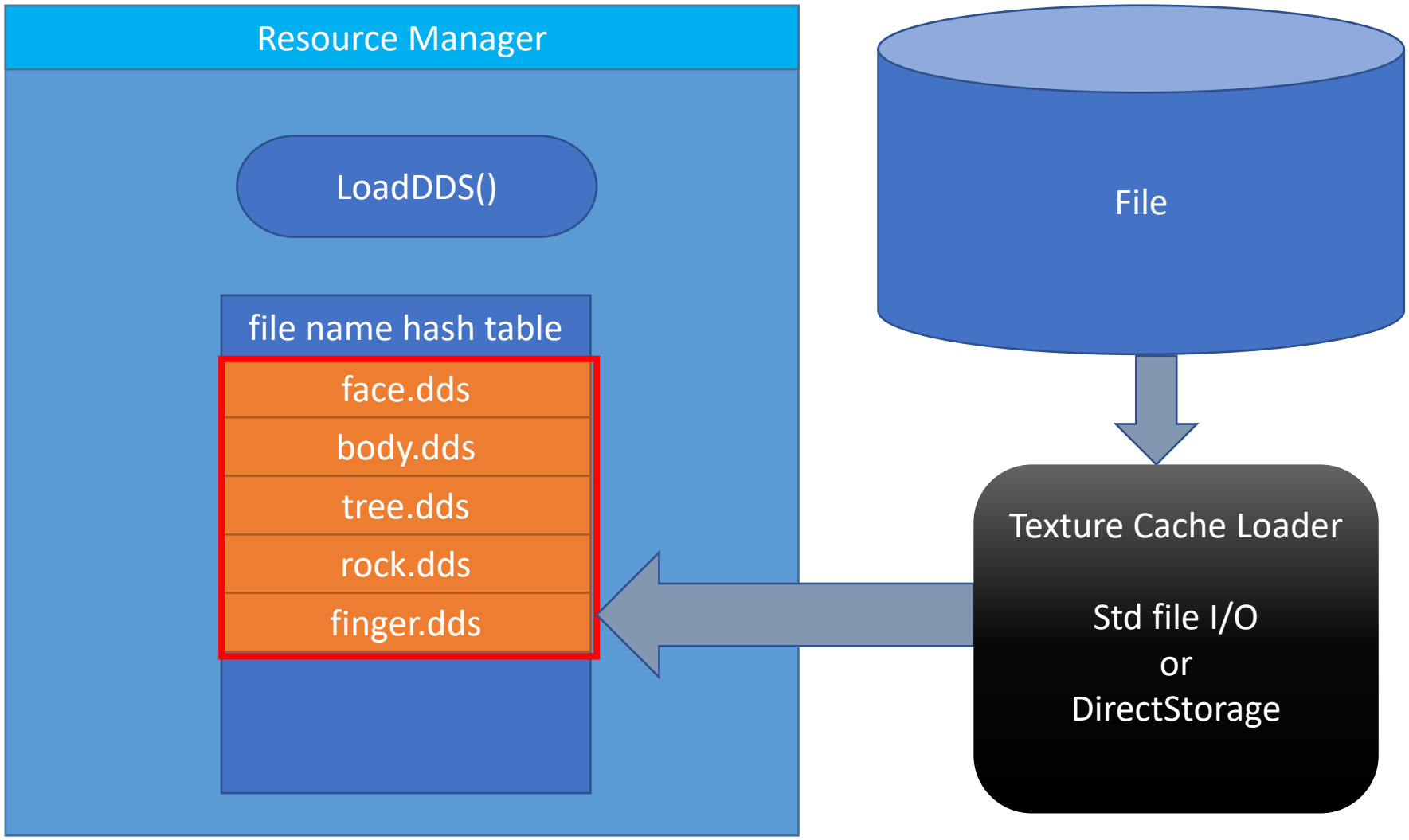


Hash table에 없으면 파일에서 읽고 hash table에 등록.

텍스처 캐시로 적용

- 텍스처 캐시는 DirectStorage를 사용하지 않아도 구현할 수 있다.
- DirectStorage를 사용하면 최적화된 I/O성능을 제공하므로 텍스처 캐시의 로딩 시간을 줄일 수 있다.
- 텍스처 캐시로 사용되는 GPU메모리가 낭비된다.
- 그에 비해 성능 향샹은 미비할 수 있다. 전체 로딩 시간 중 텍스처 로딩이 얼마나 차지하는지가 중요.

Texture Manager

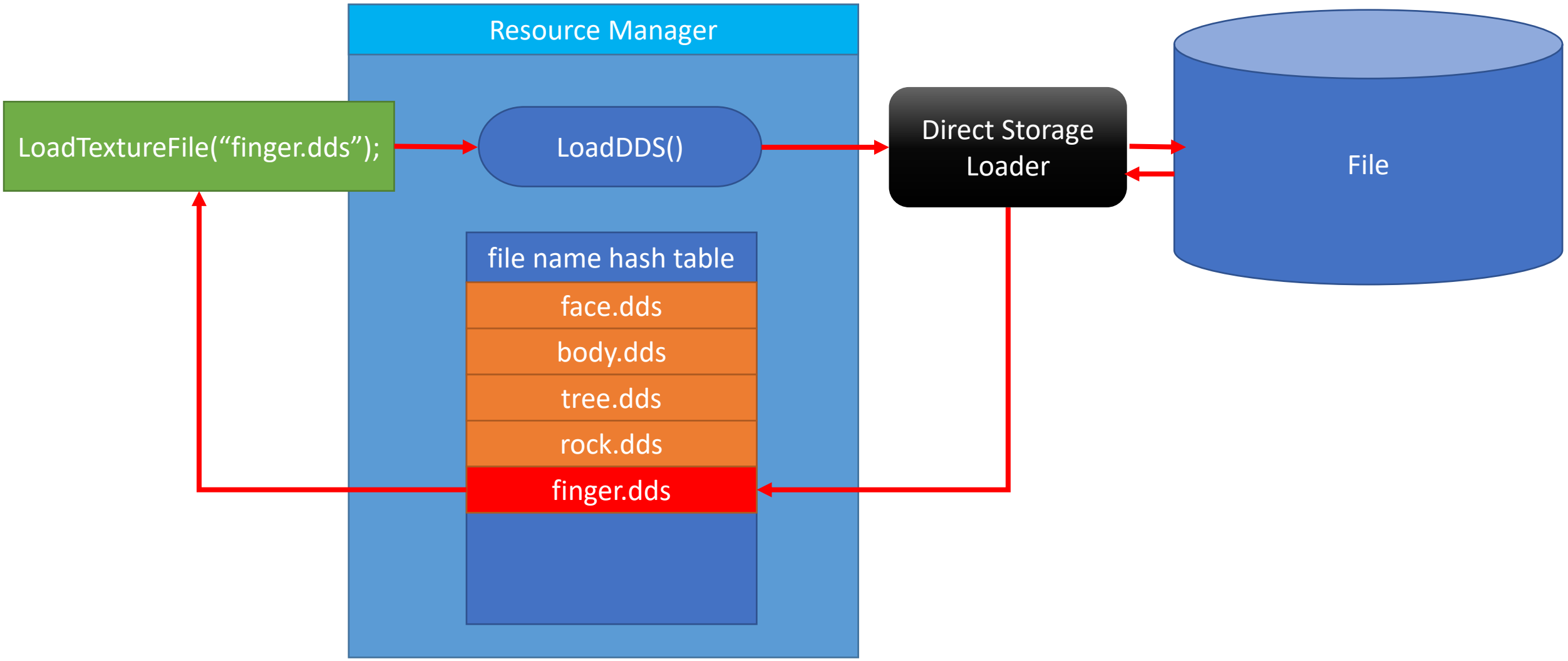


자주 사용하는 텍스처들을 초기화시 왕창 읽어서 미리 file name hash table에 등록해 둔다.
Std file I/O 대신 DirectStorage를 사용하면 초기 텍스처 캐시 로딩 시간이 짧아진다.

기존 텍스처 로딩 함수를 대체

- 텍스처 로딩 함수 호출시 `IDStorageQueue`에 enqueue만 해둔다.
- 게임 루프중에 Submit/Wait 한다. 또는 enqueue중 일정 개수마다 submit/wait한다(내부적으로 Queue의 대기열이 절반 이상 차면 자동으로 submit한다고 한다).
- 프로그래밍의 편리함 측면에서는 DirectStorage를 사용하는 쪽이 기존 방식보다 편할 수 있다.
  - D3D12에서는 DirectStorage를 사용하지 않는 기존 스타일이 프로그래밍하기에 더 복잡하다.
- 대용량의 텍스처가 엄청나게 많은 경우가 아니면 실질적으로 성능 차이를 체감하기 어렵다.

Texture Manager



개별 텍스처 로딩 시 DirectStorage를 사용한다.
로딩함수 호출시에는 enqueue만 해 두고 일정 개수 씩 모아서 submit/wait한다. 개별 로딩 시간 단축.

성능 비교 – texture cache 로딩 시간

(288MB textures)

	std file I/O(Sync-Upload Buffer)	std File I/O(Async-upload buffer)	DirectStorage	
i7 8700K , SSD 970EVO+ RAID	1061.19 ms	925.92 ms	681.03 ms	+26%
EPIC 7v12 64 (Azure VM, HDD)	12861.29 ms	11613.92 ms	10846.57 ms	+6%
Surface Book 1(SSD)	2012.39 ms	1910.86 ms	1206.59 ms	+36%

결론

- 액면 그대로 'Storage-> GPU 메모리' 로딩만 수행할 경우 분명한 성능 향상이 있다.(하지만 극적인 성능 향상은 아니다.)
- 게임에서의 Loading이라고 하는 표현은 오만가지 종류의 작업을 포함한다. 따라서 텍스처 로딩 속도가 빨라진다고 해서 로딩 속도가 크게 향상된다고 보기는 어렵다.
- 충분한 성능을 가진 PC에서는 더더욱 성능 향상이 적다.
- D3D12에서 '파일 ->Upload Buffer -> GPU리소스' 로 전송하는 단계를 비동기적으로 처리하려면 상당히 빡시다. 따라서 이렇게 처리하느니 Dstorage를 사용하는 편이 나을수도 있다(프로그래밍 편의상).
- Sparse Virtual Texturing를 사용할 경우는 큰 성능향상이 있을지도 모른다(구현 안해봐서 확신할 수 없음)

결론

- 그래도 응용해볼 여지는 많이 있으니 추후 성공적인 사례가 나올지도? 열심히 시도해 보세요.

참고자료

- microsoft/DirectStorage: DirectStorage for Windows is an API that allows game developers to unlock the full potential of high speed NVMe drives for loading game assets. (github.com)
- DirectStorage/DeveloperGuidance.md at main · microsoft/DirectStorage (github.com)
- D3D12 리소스 관리 전략(https://youtu.be/lHlAloRa-HI)