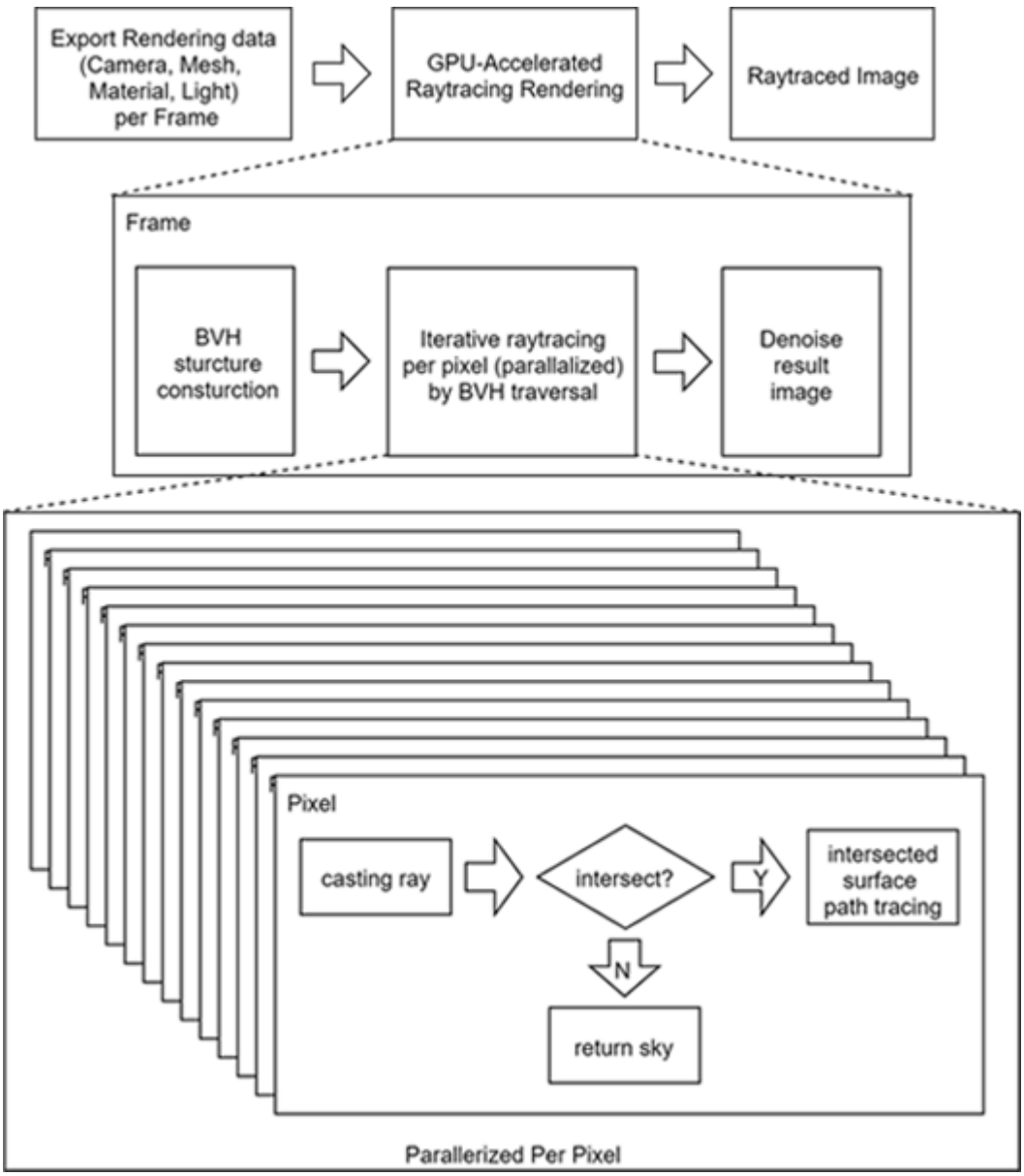
GPU-Accelerated Raytracing Renderer for Unity(Plugin)

real-time rendering framework / game engine 에서 씬 렌더링 데이터를(Camera, Mesh, Material, Light) 가져와 해당 씬을 GPGPU(CUDA/OpenCL) 혹은 GPU의 driver API(DirectX/HLSL) 를 사용하여 raytracing 기법으로 photorealistic image 를 만듦.

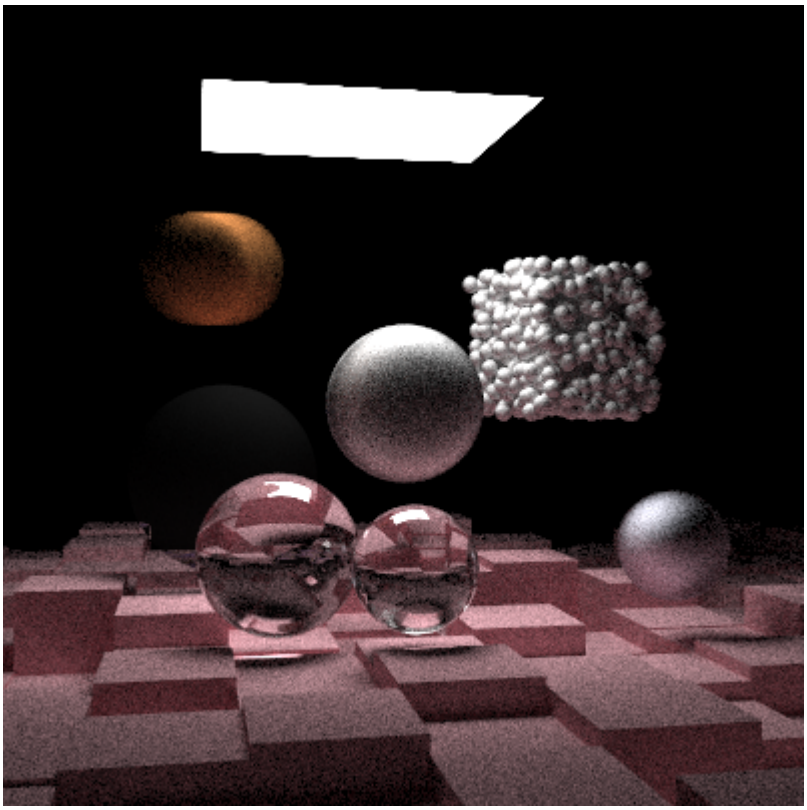
활용 방안

1. real-time rendering 의 결과인 synthesized image 와 비교하여 reference image 로써 사용되거나,(Octane Renderer at Unity)
2. 연속적으로 이미지를 생성하여 video 로 결과를 만들어 낼 수 있음. (UE4: Reflection Real-Time Ray Tracing Demo) 흔히 사용되는 GPGPU 렌더러의 역할을 함.

개략적인 구조도



데모



팀 구성

김수혁, 김한상, 정지윤

Require Techiniques

기술은 Octane Render 의 방법을 모방.(CUDA, Metropolis Light Transport)

- Raytracing
    - 각종 알고리즘에 대한 학습, 구현.
        - 학습 : metropolis light transport => 진행 중
        - 구현 : metropolis light transport -> CPU(C/C++)
        - 구현 : metropolis light transport -> GPU(CUDA)
- CUDA
    - 기본적인 programming interface 와 일반적인 optimizaton 전략에 대해서는 파악했으나 global memory fetch 가 잦기 때문에 이에 걸맞는 성능 향상 방안 필요.(GPGPU-Optimization)
    - cuda stream, graph 학습
    - how to BVH traversal ?
- Unity : rendering data-transfer?
    - single frame : light+transformation, material, mesh/mesh-per-object transformation

    - multi frame : single frame * frame count, delay time
    - 프로세스를 분리할 것인가? 자식 형태로 구성할 것인가?

- 현재 진행된 것들

    - 구현 : CPU 기반 raytracing(데모)
    - 구현 : CUDA 기반 raytracing(데모) => faster
    - 학습 : MCMC, metropolis-hastings algorithm

References

- Ray Tracing: One Weekend
- Ray Tracing: The Next Weekend
- Ray Tracing: The Rest of Your Life
- Physically Based Rendering From Theory To Implementation
- Metropolis Light Transport
- Accelerated Ray Tracing in One Weekend in CUDA
- 대규모 병렬 프로세서 프로그래밍 : CUDA를 이용한 실용적 접근