Journal of Digital Art Engineering & Multimedia Vol.9, No.1, March (2022), pp.1-12
ISSN 2508-9099(p) 2672-1198(e) *http://dx.doi.org/10.29056/jdaem.2022.03.01*

게임엔진 기반 실시간 렌더링 제작 도구 분석

Analysis of Game Engine Based Real-Time Rendering Methods

김금영[1]

Gumyoung Kim[1]

요 약

3D 기법의 영상 제작 과정은 영상을 시각화하는 렌더링 과정을 거치게 된다. 렌더링에는 많은 시간이 소요되어 대부분의 애니메이션 회사나 영화 프로덕션에서는 렌더팜(Render Farm)이라는 고가의 렌더링 서버를 갖추거나 렌트하고 있다. 이 과정은 제작 시간과 작업비용의 부담이 커서 렌더 비용을 줄이는 것이 중요한 이슈로 대두되고 있다. 최근 기술의 발달로 실시간 게임엔진을 사용해 긴 시간을 소모하는 렌더링 과정이 없이 실시간으로 결과물을 확인하면서 3D 영상을 제작하는 것이 가능해졌다. 본 연구는 실시간렌더엔진인 게임엔진의 렌더링을 영상 제작에 접목해 렌더 시간을 줄여 기존의 제작 시간을 단축하는 방법을 제시하고자 한다. 본 연구의 렌더링 실험을 위해 실내 모델링을 3D 제작 프로그램인 마야(Maya)와 게임엔진인 언리얼 엔진(Unreal Engine)에서 렌더링을 비교 분석했다. 분석 결과, 언리얼 엔진은 실시간으로 렌더되어 렌더 비용을 줄일 수 있고, 렌더된 이미지의 퀄러티 또한 마야에서 렌더된 이미지와 유사한 결과를 나타냈다. 이번 실험을 통해 언리얼 엔진의 렌더링을 마야와 비교해 분석해 봄으로써 기존 제작방식의 제작시간을 단축하는 방법을 제시하고 사용자가 실시간 렌더엔진에 접근할 수 있는 가이드를 제공한다.

핵심어 : 라이팅과 렌더링, 실시간렌더링, 게임엔진, 3D 제작 파이프라인, 오프라인 렌더링

Abstract

Image production using three-dimensional (3D) programs undergoes a process called rendering to visualize 3D data. Most animation companies and movie production companies set up or rent a costly render farm. Because this process is time-consuming and costly, the reduction of rendering cost has emerged as an important problem that requires resolution. With the recent development of technology, it has become possible to produce 3D images by checking the results in real time using real-time game engines. This work aims to overcome the limitations of the current 3D image production pipeline and propose a method for reducing the production time by adopting a game engine for real-time rendering. In the experiment conducted in this study, rendering using Maya (a 3D production program) and Unreal Engine were compared and analyzed. The analysis results indicate that Unreal Engine enables rendering in real time; consequently, the rendering cost is reduced. Moreover, the quality of the rendered image is similar to that produced by Maya. The proposed technique involves reducing the render time and providing guidance through access to a real-time rendering engine.

Keyword : Lighting and Rendering, Real-Time Rendering, Game Engine, 3D Production Pipeline, Offline Rendering

1 Department Multimedia, Seowon University, Cheongju, Korea [Professor]
e-mail: kimky_j@hotmail.co.kr

Received(January 25, 2022), Review Result(1st: February 18, 2022), Accepted(March 17, 2022), Published(March 31, 2022)

1. 서론

영화의 VFX나 3D 애니메이션과 같이 3D 프로그램을 활용한 방식의 영상제작 과정은 3D 데이터를 시각화하는 렌더링(Rendering)을 한다. 렌더링에는 많은 시간이 소요되어 대부분의 영상 프로덕션이나 애니메이션 스튜디오에서는 3D 데이터를 시퀀스(Sequence) 이미지로 렌더하기 위해 렌더팜(Render Farm)이라는 고가의 렌더링 서버[1]를 구비하고 있다. 렌더링은 프로덕션 파이프라인에서 가장 많은 시간이 필요하고, 렌더 타임을 줄이는 것은 작업비용과 밀접한 관계가 있어 국내외 많은 영상제작업체는 렌더를 실시간으로 하는 게임엔진의 도입을 시도하고 있다. 과학적인 시각화, 컴퓨터 애니메이션, 버추얼 리얼리티(Virtual Reality)는 현대 컴퓨터 그래픽의 3대 주요 논쟁거리이며, 사실적인 3D 실시간 렌더링은 이들의 핵심 기술로, 유니티(Unity)나 언리얼엔진(Unreal Engine)과 같은 실시간 렌더엔진은 카툰, 게임, 필름, 버추얼 리얼리티, 공학 영역 등 다양한 분야에 광범위하게 적용되고 있다 [2].

기존 렌더방식의(이하 오프라인 렌더링) 테크닉을 가지고 복잡한 디자인에 변형을 가하려면, 다시 렌더링을 하는데 각 반복 사이클마다 수 분, 수 시간, 심지어 수일까지 걸릴 수 있다. 또한 이 방법으로 제작된 애셋(asset)은 초기에 의도된 용도로만 사용할 수 있다. 하지만 실시간렌더방식의 애셋을 제작할 경우, 작업자는 이것을 다양한 용도와 목적으로 사용할 수 있다 [3]. 본 연구는 게임엔진을 이용하여 3D영상을 제작해 봄으로써 기존 3D영상 제작 파이프라인이 가지고 있는 한계점을 개선하고 실시간 렌더엔진인 게임엔진의 라이팅·렌더링을 영상 제작에 접목해 렌더 시간을 줄여 기존의 제작 시간을 단축하는 방법을 제시하고자 한다.

본 연구의 구성은 2절에서는 오프라인렌더링 작업방식과 실시간렌더방식의 비교를 위해 스탠포드대학 랩에서 제공하는 드래곤 모델링으로 3D 프로그램 마야(Maya)와 언리얼 엔진의 라이팅과 렌더링의 기본적인 특징과 렌더 결과를 분석했다. 3절에서는 드래곤 모델링으로 분석된 렌더링 방식을 바탕으로 복잡한 구조의 실내 모델링을 이용해 다양한 라이팅·렌더링 기능과 특징, 물리적 환경 등을 분석해 작업 시간과 렌더 시간을 오프라인 렌더방식에 비해 얼마나 줄일 수 있는지, 결과물의 노이즈 정도, 기존 방식과 유사한 렌더링 퀄러티를 도출할 수 있는지를 확인하고, 4절에서는 연구의 결론 및 향후 계획을 제시한다.

2. 오프라인렌더와 실시간렌더의 라이팅·렌더링 이론적 배경

2.1 오프라인 렌더링 3D 프로그램 마야 (Maya)

마야는 AutoDesk에서 만든 3D 프로그램으로 영화, TV 및 게임을 위한 3D 애니메이션, 모델링,

시뮬레이션 및 렌더링 소프트웨어이다. 겨울왕국, 매트릭스, 스파이더맨, 아바타, 뽀로로 등 국내외 대부분의 영상 프로젝트에 사용되고 있으며, 모델링에서 애니메이션, 시뮬레이션, 라이팅, 렌더링까지 영상 제작과정 중 프로덕션 과정에 필요한 모든 기능을 담고 있다. 마야는 영상제작에 관련한 광범위한 기능을 구현하고 있는데, 그중 마야의 기본 렌더러로 마야소프트웨어(maya software)와 아놀드(arnold) 렌더러가 있다. 마야소프트웨어 렌더러는 글로벌일루미네이션(Globla Illumination, GI), 레이 트레이싱(Ray Tracing)을 지원하지 않고, 조절할 수 있는 속성이 한정적인 렌더러로 렌더 속도는 빠른 편이다. 반면 아놀드 렌더러는 글로벌일루미네이션, 레이 트레이싱, 반사, 굴절 등의 고난도 렌더링을 정확하게 구사한다. 단점은 렌더타임이 오래 걸린다.

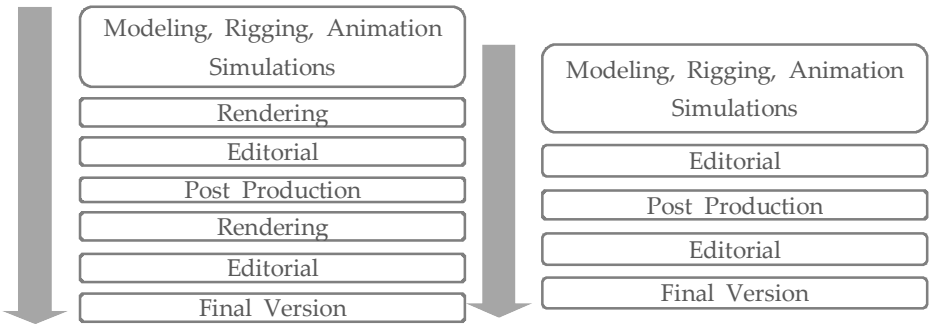
마야를 활용한 3D 영상 제작 파이프라인은 크게 3단계로 나누어 진행된다. 파이프라인이란 생산을 위해 각 구성요소가 서로 일정한 관계를 형성하는 하나의 전체를 말하며, 제작의 일괄적인 흐름이나 인력의 배치 및 구조, 이를 지원하는 모든 하드웨어적이고 소프트웨어적인 설비와 기자재 등을 통칭하는 것이다 [4]. [그림 1] 왼쪽의 3D 영상 파이프라인 중 프리프로덕션(Pre-Production)단계에서는 스토리텔링을 위한 스크립트와 스토리보드, 컨셉아트를 제작한다. 다음으로 메인 프로덕션(Main-Production)에서는 촬영기법, 세트 디자인, 조명, 캐릭터 애니메이션 연출을 위한 레이아웃 설정과 캐릭터 및 배경 모델링, 그리고 애니메이션을 제작한다. 이후 쉐이딩, 맵핑과 라이팅을 거쳐 애니메이션 연출을 완료한다. 포스트 프로덕션(Post-Production)단계는 렌더링 과정을 통해 각 프레임마다 이미지를 추출하는 과정과 영상을 편집하는 과정으로 구성된다 [5].

2.2 실시간 렌더링 게임엔진 언리얼 엔진 (Unreal Engine)

가상현실, 증강현실, 인터렉션 미디어 아트, 디지털 트윈과 같이 게임이 아닌 다양한 분야에서 게임엔진 사용이 급증하고 있다. 게임엔진은 애니메이션, 디자인 또는 그래픽과 같은 시각화를 즉시 생성하고, 렌더링 시간을 기다리지 않고 디자인을 실시간으로 수정 및 결과물을 즉시 확인할 수 있어 실시간렌더엔진 [6]이라고도 불린다. 유니티 엔진(Unity 3D)과 언리얼 엔진(Unreal Engine)은 대표적인 게임엔진으로 여러 분야에서 저작도구로 사용하고 있다.

그 중 언리얼 엔진(Unreal Engine)은 미국의 에픽 게임즈에서 개발한 3차원 게임엔진으로, 2021년 Unreal 5.0을 발표하고 루멘, 나나이트와 같은 기술을 투입해 방대한 양의 메쉬(mesh)를 계산하여 이미지를 구현할 수 있는 저작도구를 선보였다. 에픽 게임즈는 이전부터 언리얼을 영화 제작에 사용하였으며 The Beyond(2018), The Mandalorian(2019) 및 Ford v Ferrari(2019), 승리호(2021) 등 언리얼을 파이프라인에 활용한 사례가 증가하고 있다. 언리얼의 기능 중 하나인 가상 스튜디오(Virtual Studio)를 통해 이전에는 촬영 후 합성 편집과 렌더링을 진행하던 과정이 촬영과 동시에 합성과 편집이 이루어져 중간 결과물을 감독이 바로 확인할 수 있으며, 최종 렌더링 또한 훨씬 시간이 단축되어 진행되고 있다 [7].

게임엔진 기반의 3D영상 제작공정(Pipeline)은 프리 프로덕션단계는 기존 파이프라인과 동일하지만, 메인 프로덕션 단계에서 차이를 보인다. [그림 1]의 오른쪽 언리얼 제작과정을 보면, 3D 그래픽 툴로 제작된 캐릭터와 배경 모델링, 애니메이션 데이터는 게임엔진으로 옮겨진다. 이후 작업은 게임엔진에서 이루어지며 연출환경구성과 레이아웃 단계에서 에셋(Asset) 배치, 라이팅, 머티리얼(Material)을 적용한다. 연출단계에서는 스토리보드를 참조로 게임엔진 내 시네마틱 카메라를 설정한다. 카메라의 애니메이팅 연출단계를 거친 후, 실시간으로 재생되는 장면을 추출하고 편집한다. 작업의 스토리보드 단계를 건너뛰고 게임엔진을 사용하게 되면 처음부터 컴포지팅(Compositing) 작업이 진행되어 렌더팜을 사용하지 않고 완성이 가능해진다. 이처럼 3D영상 제작공정에서는 렌더링을 통해 결과물을 수시로 확인해야 하지만, 게임엔진 기반 영상작업은 실시간으로 제작된 영상을 바로 확인할 수 있어 피드백 반영시간을 절약할 수 있다 [5][7].



[그림 1] 마야 작업공정(왼쪽)과 언리얼 작업공정(오른쪽) 비교

[Fig. 1] Artist Workflows of Maya(left) and Unreal Engine(right)

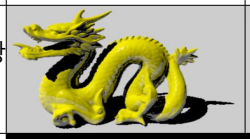
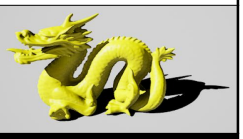
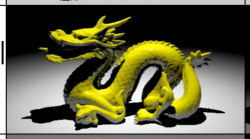
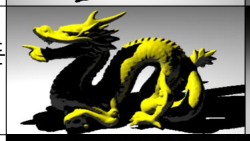
2.3 마야와 언리얼 엔진의 라이팅 종류

본 연구는 라이팅·렌더링 기법을 마야와 언리얼 엔진에서 비교 분석했다. 렌더링은 라이팅과 밀접한 관계가 있다. 화면 구성 내에 설치한 라이팅은 3D에셋과 연산 과정을 통해 렌더링 이미지로 추출되며 작업물을 확인하기 위해 시간이 소요된다. 반면 게임엔진에 설치한 라이팅은 실시간으로 최종 이미지를 확인하고 수정이 있을 시 피드백을 바로 진행한다.

마야의 라이트 종류는 [표 1]에 정리된 바와 같이 디렉셔널(Directional) 라이트, 스팟(Spot) 라이트, 포인트(Point) 라이트, 에어리어(Area) 라이트, 메쉬(Mesh) 라이트, 스카이 돔(Skydome) 라이트로 각기 다른 속성을 가지고 있다. 언리얼 엔진의 라이트 유형은 Directional, Point, Spot, Sky, Rect 라이트로 기본적인 속성은 마야의 라이트와 유사하다. 디렉셔널 라이트는 야외 주광원에 주로 사용된다. 포인트 라이트는 고전적인 '전구'같은 라이트로, 한 지점에서 모든 방향으로 빛을 뿜는다. 스

팟 라이트 역시 한 지점에서 빛을 뿜으나, 한 쌍의 원뿔로 그 빛이 제한된다. 스카이 라이트는 씬의 배경을 캡처하여 레벨의 메시에 라이팅으로 적용한다. 마야와 언리얼 엔진의 공통된 라이트 속성은 라이트의 컬러, 빛의 강도, 그림자의 부드러움 정도, 그림자의 노이즈를 조절하는 샘플을 조절할 수 있는 기능이 있다. 각 라이트는 씬의 크기와 위치, 종류, 그리고 시간대와 분위기에 따라 다르게 세팅이 되는데, 라이트의 종류와 특성을 파악하고 프로젝트의 목적에 맞게 선택하는 것이 중요하다. 본 연구는 마야와 언리얼 엔진에 공통으로 있는 디렉셔널 라이트, 스팟 라이트, 포인트 라이트, 렉트 라이트, 스카이 라이트, 총 5가지 라이트를 비교하여 [표 1]로 정리했다.

[표 1] 마야와 언리얼 엔진의 라이트 종류
[Table 1] Light Types in Maya and Unreal Engine

Type	특징	Maya	Unreal Engine
Directional	· 태양광과 같이 무한히 먼 거리에서 오는 광원을 표현		
Spot	· 손전등과 같이 방향성을 갖고 일정 영역에 빛을 비추는 조명		
Point	· 텅스텐 전구와 같이 광원을 중심으로 모든 방향으로 발산하는 빛		
Area/Rect	· 직사각 모양의 빛 · 텔레비전이나 모니터 화면, 벽 등처럼 직사각 영역을 비추는 광원 시뮬레이션에 사용		
Sky	· 하늘에서 오는 빛을 재현하는 조명 · 주로 HDRI(High Dynamic Range Image)를 color에 연결해 광원으로 사용		

3. 실시간 렌더링과 오프라인 렌더링의 비교 실험

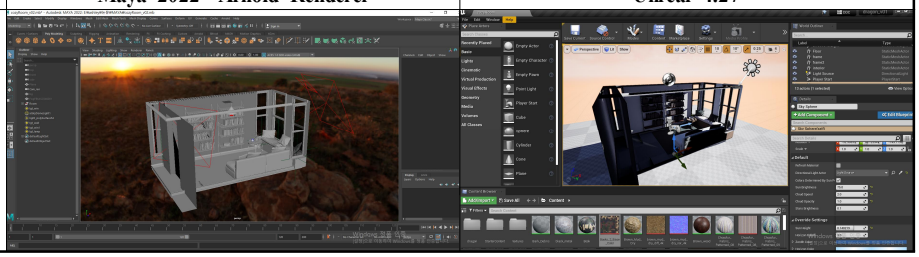
3.1 실험 환경

드래곤 모델링으로 분석된 라이팅·렌더링 방식을 바탕으로 복잡한 구조의 실내 모델링을 이용해 작업 시간과 렌더 시간을 오프라인 렌더방식에 비해 얼마나 줄일 수 있는지, 결과물의 노이즈 정도와 오프라인 렌더와 유사한 렌더링 퀄러티를 도출할 수 있는지 실험했다. 본 연구의 하드웨어 및 소프트웨어 실험 환경은 CPU Intel i7, GPU geforce 3050, Memory 32MB 환경에서 실내 배경 모

델링을 사용했다. 소프트웨어는 마야 2022, Arnold 3.2.2 버전을, Unreal Engine 4.27 버전을 사용했다. [표 2]는 각 프로그램의 작업공간에 관한 표이다.

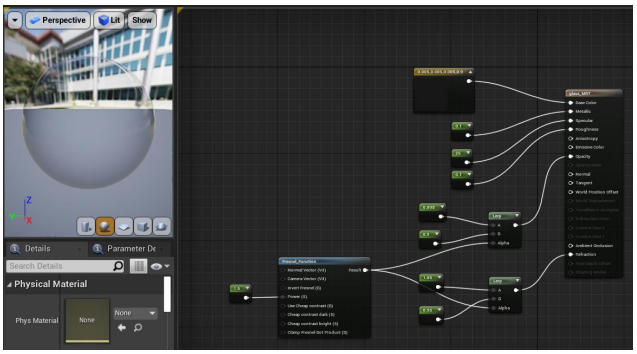
[표 2] 마야와 언리얼 엔진의 작업공간

[Table 2] Workspace of Maya and Unreal Engine

Program	Maya 2022- Arnold Renderer	Unreal 4.27
workspace		

3.2 쉐이딩과 텍스춰링

마야에서 모델링된 데이터를 언리얼 엔진으로 불러와 씬 세팅을 시작했다. 불러온 오브젝트의 재질은 언리얼 엔진의 블루프린트(Blueprint)로 재작업 했다. [그림 2]의 유리 텍스춰와 같이 블루프린트에서 Shader의 Reflection, Roughness, Specular를 조절해 모델링에 반사와 투명도 등의 디테일을 더했다.



[그림 2] 언리얼 엔진 블루프린트(오른쪽)와 유리 쉐이더 썸네일(왼쪽)

[Fig. 2] Shader structure in Blueprint(right) and Glass shader in Unreal Engine(left)

언리얼 엔진의 비주얼 스크립팅 시스템인 블루프린트는 언리얼 에디터 안에서 노드 기반 인터페이스를 사용하여 게임플레이 요소를 만드는 개념을 토대로 한 비주얼 스크립팅 시스템이다. 일반적인 스크립팅 언어와 마찬가지로, 엔진 내 객체 지향형 클래스 또는 오브젝트를 정의하는 데 사용된다. 기본적인 형태의 블루프린트는 시각적인 스크립팅으로 게임에 추가되는 것으로 노드, 이

벤트, 함수, 변수 등을 선으로 연결하여 복잡한 게임플레이 요소를 만드는 것이 가능하다 [8].

3.3 라이팅과 렌더링

3.3.1 키 라이트 세팅

이번 실험의 시간대는 노을진 배경으로 노을빛을 받아 창문 사이로 은은하게 오렌지색 햇빛이 비치는 컨셉으로 설정했다. 조명을 설치할 때 가장 먼저 주요하게 설치해야 하는 것이 키 라이트로, 우선 노을빛을 구사하기 위해 [그림 3]과 같이 Directional Light를 키 라이트로 세팅했다. Directional Light는 패러렐(Parallel) 라이트로 아주 넓은 공간까지 뻗어가는 평행 성질을 가진 라이트이다. 달빛이나 태양빛을 구사하는데 주로 쓰인다. Directional Light의 Intensity, 컬러 등을 조절해 노을 시간대 분위기를 만들어 주었다.

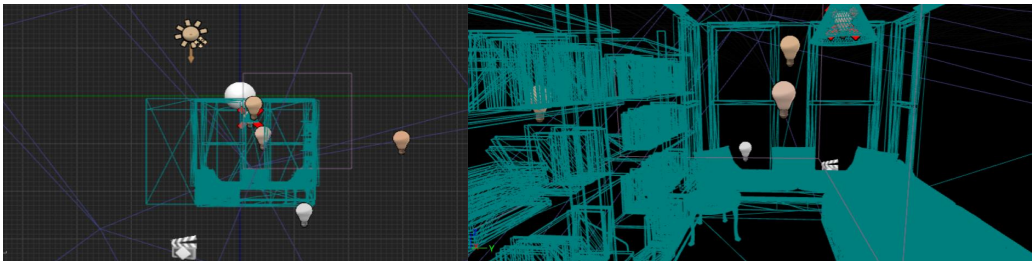
움직임이 없는 정적인 오브젝트에 배치된 빛의 경로를 계산하여 원본 텍스처에 추가로 베이크된 텍스처를 더하고 화면에 빛을 표현하는 방식을 사용하는데, 이를 라이트맵핑(Lightmapping) [9]이라고 한다. 라이트맵을 활용하면 하드웨어에 가중되는 부담을 줄일 수 있다. 라이트맵핑은 고정된 조명을 맵핑으로 출력함으로써 실제 조명은 아니지만, 조명과 같은 효과를 줄 수 있다. 언리얼 엔진에서는 레벨에 라이트 또는 오브젝트를 추가 혹은 이동시킬 때마다 라이트맵을 리빌드(Re-Build)해 줘야 정확한 표현을 확인할 수 있다.

실험 결과 라이트의 빌드타임이 1분 정도 소요되었는데, 실제 프로덕션에 적용되는 복잡한 구조의 모델링을 사용하게 되면 빌드 타임이 크게 차이 날 것으로 예상된다. 결과물의 그림자 해상도가 깨지는 현상이 생겨 Directional Light 옵션의 Cascade Shadow map 활성화하고 Dynamic Shadow Distance Static 값을 높게 조절, Num Dynamic Shadow Cascades을 4로 올려 선명하게 조절했다.

[그림 3] 키 라이트로 디렉셔널 라이트가 적용된 실내 배경
[Fig. 3] Directional light applied in the room as a key light

3.3.2 추가적인 라이팅 세팅

키 라이트가 설정되면, 실내에 있는 조명 소스들을 파악하고 그에 맞추어 추가적인 라이트들을 설치하게 된다. 이번 테스트는 노을이므로 [그림 4]와 같이 창 반대편 벽 쪽에 필 라이트(Fill light)를, 창 옆쪽 렉트 라이트를 추가해 소파와 쿠션의 가장자리에 노을빛을 더해주고, 램프와 같은 조명 소스들을 활용해 은은한 느낌의 광원들을 설치해 주었다.



[그림 4] 키 라이트와 필 라이트 적용된 실내 배경 와이어 프레임 모드

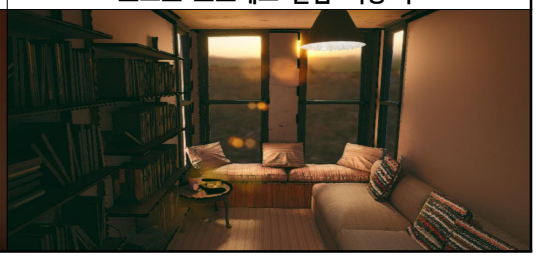
[Fig. 4] Additional Lights applied in Unreal engine Wire Frame Mode

3.3.3 포스트 프로세싱과 최종 렌더링

마지막 단계로 언리얼 엔진에서 포스트 프로세싱(Post Processing)을 적용했다. 마야에서는 렌더링 후 누크(Nuke)와 같은 전문 합성 프로그램에서 합성작업을 거쳐 최종 결과물을 완성하게 되지만, 언리얼 엔진에서는 합성작업을 대체할 만한 툴로 포스트 프로세싱을 활용할 수 있다. 포스트 프로세싱은 기존에 렌더링 된 씬에 합성 효과를 더하는 작업이다. 이 Post Process 효과를 적용하기 위해 언리얼에서 제공하는 Post Process volume을 사용하는데, 렌즈 플레어, 화면 색조 및 블러링 같은 효과를 낼 수도 있고, 볼륨끼리 또는 플레이어와의 상호작용 방식을 정의한다 [8]. 이번 실험에서는 [표 3]과 같이 포스트 프로세싱 옵션 중 Color Grading으로 따뜻한 색감을 더하고 offset으로 책장의 어두운 영역을 밝게 올려주고 창가에서 오는 노을빛과 대치되도록 차가운 기운을 더했다. Bloom으로 Glow 이펙트를, Vignetting으로 화면 가장자리를 살짝 어둡게 눌러주어 필름 효과를 주었다.

라이트 세팅이 끝나게 되면 마야에서는 이미지를 exr포맷으로 시퀀스 렌더해 Nuke나 After effect 같은 전문 합성 프로그램에서 합성하는 과정이 필요하다. 이에 비해 언리얼은 실시간 재생 및 동영상 출력도 가능해지면서 렌더팜을 사용하지 않고도 결과물의 완성이 가능했다. 물론 폴리곤을 훨씬 많이 사용한 영상용 폴리곤 모델링과는 차이를 보이는 한계점이 존재하지만, 지속적인 하드웨어의 발전으로 해결이 가능할 것으로 보인다.

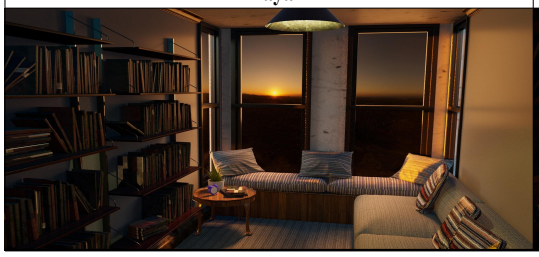
[표 3] 언리얼 엔진의 포스트 프로세스 볼륨 적용 전과 적용 후
[Table 3] Before and After Post Process Volume

포스트 프로세스 볼륨 적용 전	포스트 프로세스 볼륨 적용 후
	

3.4 라이팅과 렌더링 실험 결과

게임엔진은 복잡한 툴로 오프라인 렌더링 방식과 비교하는 것은 까다로운 일이지만 같은 프로젝트를 두 플랫폼에 적용한 비교는 가능할 것이다. 혹은 측정이 가능한 기준으로 두 플랫폼을 객관적으로 비교할 수도 있다. 정확한 비교를 위해 적정한 복잡도의 프로젝트를 비슷한 방법으로 양쪽 모두의 플랫폼에 적용하는 것이 중요하다 [10]. 3D 마야 프로그램과 언리얼 엔진을 활용한 라이팅·렌더링 실험 결과, 두 프로그램 모두 비슷한 라이팅 타입과 속성을 가지고 있어 [표 4]와 같이 근접한 퀄러티의 결과물을 만들어 낼 수 있었다.

[표 4] 마야와 언리얼 엔진 최종 결과물 비교
[Table 4] Final Render Image Comparison

Maya	Unreal Engine
	

[표 5]에 정리된 바와 같이 마야에서는 실내 배경에 총 6개의 라이트를 세팅했는데, Directional light로 창에서 들어오는 노을빛을, 4개의 Area light를 rectangle shape으로 창가, 천장과 벽의 필 라이트로, 1개 mesh 라이트를 전등에 설치해 전구처럼 발광하도록 세팅했다. 쉐이더는 Blinn을 사용해 Specular를 더했다. 렌더 샘플은 노이즈가 생기지 않는 프로덕션 퀄러티로 렌더했을 때, 한 프레임당 렌더 타임은 18분 41초가 걸렸다. 라이트를 세팅한 작업 시간은 약 1시간이 소요되었다.

[표 5] 라이팅과 렌더링 실험 결과

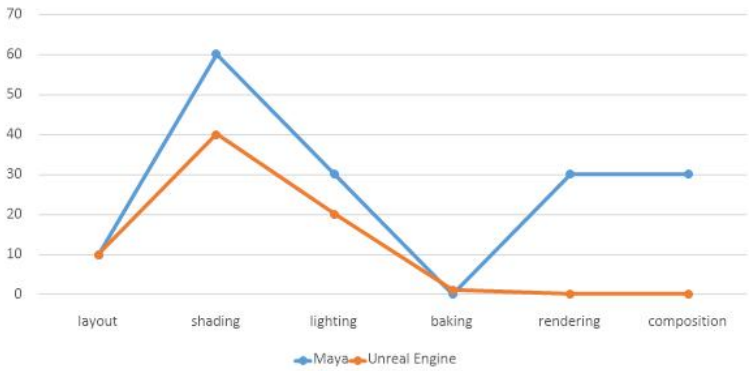
[Table 5] Comparison of Lighting and Rendering Results

Item		Maya	Unreal Engine
라이트 종류		1 SkyDome light, 4 Area lights, 1 Mesh light	1 Sky, 1 Directional , 4 Rect, 1 Point light
총 사용된 라이트 갯수		6 개	7 개
쉐이더		Blinn	블루 프린트
렌더 샘플 세팅		Camera(AA) 5, Diffuse 4, Specular 4, Transmission 2, SSS 2, Volume Indirect 2	Spatial Sample Count 8, Compression setting 100
작업 단계		5 단계	4 단계
작업 과정 별 작업 시간	Layout	10 분	10 분
	Shading Texturing	1 시간	40 분
	building	0 분	1 분
	Lighting	30 분	20 분
	Rendering	18 분 41초	0 분
	Composition	30 분	0 분
총 작업 시간		2시간 28분 41초	1 시간 11분

언리얼에서는 마야에서 모델링된 실내 배경을 임포트해 라이팅 작업을 시작했다. Directional light로 창에서 들어오는 노을빛을, Sky light로 환경광을, 4개의 Area light와 1개의 Point light로 실내등과 창가, 필라이트로 조명했다. 라이팅 작업 시간은 약 30분으로, 라이트를 배치할 때마다 테스트 렌더를 할 필요가 없이 바로 확인할 수 있어 마야보다 작업 시간이 절감되었다. 단지 라이트를 빌드 하는데 1분 정도의 시간이 추가되었다. 렌더링 시간은 불필요했다.

작업 프로세스상의 차이는 마야의 경우 물리 기반의 라이팅을 테스트 렌더를 거쳐 최종 결과물로 만들 수 있고, 또한 합성을 위한 라이팅 패스(pass)들을 뽑을 수 있어 합성에 유리한 점이 언리얼 엔진과의 차이점으로 드러났다. 이와 달리 언리얼 엔진은 라이팅 설치 후 빌드해야 하는 번거로움이 있으나, 빌드 시간이 수분으로 작업 시간에 크게 영향을 주지 않았다. 모델링 데이터가 큰 경우는 빌드 시간이 수십 분도 걸릴 수 있으나, 결정적으로 한번 빌드를 하면 여러 카메라 각도에서 라이팅을 확인할 수 있고, 렌더 시간이 따로 필요 없어 실시간으로 렌더되는 점이 큰 장점으로 드러났다. 언리얼은 라이팅 패스를 제한적으로 뽑을 수 있다는 단점이 있지만, 언리얼의 포스트 프로세스 볼륨을 대안으로 활용할 수 있다.

[그림 5]를 보면 쉐이딩, 렌더링, 합성에서 마야의 작업 시간이 더 많이 걸리는 것이 확연히 드러난다. 이번 실험으로 룩 개발과 직접 관련이 있는 작업과정은 언리얼이 유리한 것으로 확인되었다.



[그림 5] 마야와 언리얼 엔진의 공정별 작업 시간 비교(단위:분)

[Fig. 5] Comparison of Working Time in Maya and Unreal Engine (unit:min)

4. 결론

본 연구는 현재 3D 영상 제작 파이프라인에 게임엔진의 라이팅·렌더링을 접목해 렌더 시간을 줄여 기존의 제작 시간을 단축할 방법을 제시하고자 했다. 이를 위해 실내 배경 모델링으로 3D 마야 프로그램과 언리얼 엔진의 렌더링을 비교 분석했다. 실험 결과 마야에서 제작한 에셋을 손쉽게 언리얼로 불러올 수 있고, 라이트 세팅 또한 마야와 비슷한 속성을 가지고 있어 기존 오프라인 3D 파이프라인에 익숙한 사용자도 큰 부담 없이 접근할 수 있었다.

폴리건 수가 많은 모델링 데이터나 합성이 필요한 높은 수준의 프로젝트는 마야나 기존 3D 프로그램으로 렌더하여 합성하는 것이 퀄러티 측면에서 유리할 것으로 파악된다. 그러나 TV시리즈 같이 복잡한 합성이 필요 없는 프로젝트의 경우는 언리얼도 비슷한 수준의 결과물을 만들어내는 것을 확인했다. 분석 결과, 언리얼과 같은 실시간렌더엔진은 기존 방식과 비교하여 렌더링 비용과 시간 절감이 가능하며 렌더팜과 같은 대규모 렌더링을 할 필요가 없어 제작 비용이 절감되는 효과를 볼 수 있다는 것이 큰 장점으로 드러났다.

기존 파이프라인과의 대치, 새로운 작업환경에 적응, 게임엔진을 다루는 비게임분야 전문가 부재 등 여러 가지 어려움이 있음에도 불구하고 가까운 미래에 비게임엔진 분야의 주요 렌더러로 도입될 것을 예상하는 이유가 바로 이러한 장점 때문이다. 이번 실험을 통해 언리얼의 라이팅 세팅을 어떤 식으로 할 것인지, 마야의 라이팅·렌더링과 비교해 렌더 타임, 노이즈 퀄러티, 렌더 방식 등을 분석해 봄으로써, 기존 3D 애니메이션 파이프라인에 익숙한 아티스트들에게 실시간렌더엔진에 접근할 수 있는 가이드를 제공했다는 데에 의의가 있다. 향후 연구 계획은 언리얼 엔진과 더불어 게임엔진 분야에서 시장 점유율이 가장 높은 게임엔진인 유니티를 대상으로 두 엔진이 가진 특징들을 분석하여 렌더링을 측정해 비교 분석할 필요가 있을 것으로 판단된다.

References

[1] R. G. Ramani, I. A. Nivas, “A rendering pipeline framework for photorealistic rendering of animated virtual objects into real scenes”, 2014 International Conference on Recent Trends in Information Technology, April 10-12, 2014, Chennai, India, pp. 1-6, doi: 10.1109/ICRTIT.2014.6996125.

[2] D. Seng, H. Wang, “Realistic Real-Time Rendering of 3D Terrain Scenes Based on OpenGL”, 2009 First International Conference on Information Science and Engineering, December 26-28, 2009, Nanjing, China, pp. 2121-2124, doi: 10.1109/ICISE.2009.871.

[3] K. Pimentel, “Real time is the Future. Why do we have to change it now?”, unrealengine.com, https://www.unrealengine.com/ko/tech-blog/real-time-is-the-future-why-change-now?sessionInvalidated=true, (accessed November 28, 2021).

[4] S. S. Yang, “A study on the System Process of Production pipeline of 3D animation”, The Korea Contents Association Conference, May 1, 2008, Seoul, Korea, pp. 198-202.

[5] C. H. Jung, M. J. Kim, “Comparative Analysis for Production Pipeline Between 3D Animation and Machinima”, Journal of Korea Entertainment Industry Association, vol. 10, no. 2, April 2016, pp. 313-321, doi: 10.21184/jkeia.2016.04.10.2.313.

[6] W. Shi-an, “Research on the Real-Time Rendering of Global Illumination of the Virtual Reality System Based on Cloud Computing”, IEEE Trans. 12th International Conference on Intelligent Computation Technology and Automation (ICICTA), October 26-27, 2019, pp. 236-239, doi: 10.1109/ICICTA49267.2019.00057.

[7] D. H. Suh, “A Study of a 3D Animation Production Pipeline Using a Game Engine”, CONTENTS PLUS, vol. 19, no. 5, December 2021, pp. 71-84.

[8] B. B. Ramos, J. Doran, Unreal Engine 4 Material, Acorn Public, 2020.

[9] M. K. Kim, Tthe Game Graphics, Viel Books, 2015.

[10] A. Šmíd, “Comparison of Unity and Unreal Engine”, Bachelor's Thesis, Faculty of Electrical Engineering Department of Computer Graphics and Interaction, Web and Multimedia, Czech Technical University, Czech Republic, 2017.