*Journal of Next-generation Convergence Information Services Technology* Vol.11, No.5, October (2022), pp.499-509
*ISSN 2384-101X(p) 2672-1163(e)* *http://dx.doi.org/10.29056/jncist.2022.10.04*

# 실시간 렌더링 제작 도구 비교 분석

# Comparative Analysis of Real-Time Rendering Methods

김금영[1]

Gumyoung Kim[1]

## 요 약

전통적인 3D그래픽의 제작과정에서는 3D 데이터를 시각화하는 렌더링(Rendering)을 하게 되는데, 렌더링은 막대한 시간과 비용을 소모한다. 렌더링 결과를 실시간으로 확인할 수 있는 '실시간 렌더링' 기능의 게임엔진 활용으로 이러한 작업 기간을 크게 단축할 수 있다. 본 연구는 시장점유율이 가장 높은 게임엔진으로 다양한 분야에서 사용되고 있는 언리얼 엔진(Unreal Engine)과 유니티 엔진(Unity 3D)의 라이팅·렌더링을 비교하고, 두 엔진을 이용하여 3D 영상을 라이팅·렌더링해 3D 영상 제작 파이프라인에 게임엔진의 라이팅·렌더링을 접목할 방법을 제시하고자 한다. 본 연구는 실내 모델링을 이용해 유니티와 언리얼 엔진에서 렌더링을 비교 분석했다. 분석 결과, 언리얼 엔진과 유니티는 실시간으로 렌더가 가능해 렌더 비용을 절감할 수 있고, 렌더링 된 이미지의 완성도 또한 기존 오프라인 렌더러에서 렌더링 된 이미지와 유사한 결과를 나타냈다. 이번 실험을 통해 유니티와 언리얼 엔진의 렌더링을 비교해 분석해 봄으로써 영상 제작 시간을 단축할 수 있는 방법을 제시하고 사용자가 실시간 렌더엔진을 활용할 수 있는 가이드를 제공한다.

핵심어 : 라이팅과 렌더링, 실시간렌더링, 게임엔진, 3D영상 제작 파이프라인

## Abstract

In a traditional 3D production pipeline, image production using 3D programs undergoes a process called rendering to visualize 3D data and this process is time-consuming and costly. By adopting a game engine for real-time rendering to check render images instantly, we can reduce the production time a lot. In this experiment conducted in this study, rendering process using game engines with the highest market share in various fields, Unity and Unreal Engine were compared and analyzed. Also this work aims to propose a method for adopting a game engine lighting and rendering to 3D production pipeline by using both engines. In this study, interior modeling was used to compare rendering in both Unity and Unreal engine. The analysis results indicate that Unity and Unreal Engine enable rendering in real time; consequently, the rendering cost is reduced. Moreover, the quality of the rendered image is almost similar to that produced by offline rendering images. The proposed technique involves reducing the render time and providing guidance for artists through access to a real-time rendering engine by comparing and analyzing the rendering of Unity and Unreal Engine through this experiment.

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

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

* 본 논문은 2021년도 차세대컨버전스정보서비스학회 동계학술대회에서 발표한 논문을 수정 및 보완한 것임

*Received(August 29, 2022), Review Result(1st: September 15, 2022, 2nd: October 3, 2022), Accepted(October 14, 2022), Published(October 31, 2022)*

# 1. 서론

게임의 그래픽 기술은 처음 3D 그래픽 카드가 등장했던 1995년과 비교해도 비약적으로 발전했다. 오랫동안 게임 기술의 중요한 화두는 실사와 구분할 수 없는 실감나는 영상을 실시간으로 만드는 것이었다. 게임 그래픽의 실사적인 표현이라는 목표를 향해 하드웨어와 기술이 발달했고, 이제는 다양하고 예술적인 표현이 가능하게 되었다 [1]. 이렇게 발전된 게임 그래픽 기술은 더 이상 게임에만 한정되지 않고 애니메이션, 영화, 가상현실, 증강현실, 인터랙션 미디어 아트, 디지털 트윈과 같이 다양한 분야에서 사용이 급증하고 있다. 특히 게임을 구동시키는데 필요한 핵심 기능들을 담은 소프트웨어인 게임엔진은 그래픽을 즉시 렌더링(Rendering)하고, 렌더 시간을 기다리지 않고 그래픽을 실시간으로 확인할 수 있어 실시간렌더엔진 [2]이라 불리며, 기존 오프라인 렌더방식의 대체 기술로 각광받고 있다.

전통적인 3D그래픽의 제작과정은 3D 데이터를 시각화하는 렌더링을 하게 되는데, 렌더링은 막대한 자원을 소모한다. 예를 들어 1시간짜리 영상을 만든다면 렌더링에 걸리는 시간은 약 7년이다. 이에 지금까지는 여러 대의 컴퓨터를 합친 '렌더팜'을 구축해 작업량을 분산하는 '오프라인 렌더링'을 사용했는데, 이런 방법도 비용과 시간이 많이 든다. 만약 렌더링을 거친 결과물에서 카메라 구도나 오브젝트 배치 등을 바꾸고 싶다면, 또다시 장면을 수정하고 처음부터 렌더링을 거쳐, 엄청난 시간을 소비해야 했다. 영상 작업 효율이 크게 떨어지는 것이다. 그러나 게임 엔진을 사용하면 이러한 작업 기간을 단축시킬 수 있게 되는데, 렌더링 결과를 실시간으로 확인할 수 있는 '실시간 렌더링' 기능 때문이다 [3].

본 연구는 실시간 렌더엔진인 게임엔진 중 시장점유율이 가장 높고 여러 분야에서 사용되고 있는 언리얼 엔진(Unreal Engine)과 유니티 엔진(Unity 3D)의 라이팅·렌더링을 비교하고, 두 엔진으로 3D 영상을 제작해 3D 영상 제작 파이프라인에 게임엔진의 라이팅·렌더링을 접목할 방법을 제시하고자 한다.

본 연구의 구성은 2장에서는 오프라인 3D프로그램과 실시간렌더 프로그램의 라이팅과 렌더링을 비교하기 위해 스탠포드 대학에서 제공하는 드래곤 모델링을 사용했다. 드래곤 모델링으로 영화와 애니메이션 등에 광범위하게 사용되고 있는 3D그래픽 프로그램 마야(Maya)와 게임엔진 유니티, 언리얼 엔진의 라이팅과 렌더링의 특징과 렌더 결과를 비교 분석했다. 3장에서는 드래곤 모델링으로 분석된 렌더링 결과를 바탕으로 복잡한 구조의 실내 배경 모델링을 사용해 게임엔진의 다양한 라이팅·렌더링 특징과 기능, 물리적 환경 등을 분석했다. 작업 시간과 렌더 시간이 각각 얼마나 걸리는지, 결과물의 노이즈 정도, 렌더링 퀄러티를 비교하고, 4장에서는 연구의 결론과 향후 계획을 제시한다.

# 2. 게임엔진 라이팅과 렌더링의 이론적 배경

## 2.1 유니티 (Unity 3D)

유니티는 Unity Technologies가 2004년 8월 개발한 게임엔진으로 전체 게임 개발 시장의 62%를 차지하는 콘텐츠 제작도구이다. 국내에서도 20만명이 넘는 사용자가 유니티로 콘텐츠를 제작하고 있다. 유니티는 게임엔진으로 널리 알려졌지만, 그래픽 기능이 크게 개선되면서 애니메이션, 영화, 광고, 건축 등 광범위한 산업에 쓰이고 있다. Siggraph 2018에서 영화를 제작하기 위한 유니티의 활용에 대해 발표하며, 렌더링과 쉐이더 기술을 발전시켜 고품격 그래픽을 구현하는 데 주력하고 있다 [4].

[표 1]에서 보는 바와 같이 유니티는 복잡한 3D게임을 간단하게 만들 수 있도록 해주는 그래픽 환경의 통합형게임 개발 프로그램이다. 유니티에서 개발된 게임 소스는 안드로이드, 윈도우, iOS, 블랙베리, 리눅스, 플래시, 웹 브라우저, 플레이 스테이션, X박스 360, 윈도우폰 등 수 많은 기기로 배포된다. 유니티는 직관적이고 간단한 인터페이스로 최적의 작업 환경을 제공하며 게임 개발을 위한 강력한 도구모음을 제공하고 있다. 또한 프로그래밍 언어를 적용해 게임을 구동시킬 수 있다. 3D 모델링 임포트, 조명과 그림자, 특수효과, 오디오, 질감, 지형, 물리법칙 등을 이용해 사실적으로 영상을 표현할 수 있다 [5].

[표 1] 유니티와 언리얼 엔진 특징 비교
[Table 1] Unity and Unreal Engine Comparison

| Feature | Unity | Unreal |
|---|---|---|
| Scripting Languages | UnityScript, C# | Blueprint Visual Scripting, C++ |
| Lighting | Spot light, Directional, Point, Area light | Directional, Point, Spot, Area light, Sky light |
| Global illumination | Baked GI, Image based GI, Pre-computed realtime GI | Baked GI, Image based GI |
| Import 3D model formats | 3ds, fbx, dae, c4d, dxf, jas, lxo, blend | obj, fbx, srt (speedtree) |
| UI | IMGUI, UI objects on Canvas | Widgets and Blueprints |

## 2.2 언리얼 엔진 (Unreal Engine)

미국의 에픽 게임즈에서 개발한 언리얼 엔진은 게임엔진으로, 2021년 발표된 Unreal 5.0에서는 루멘, 나나이트와 같은 기술을 장착해 방대한 양의 데이터를 계산하여 사실적인 그래픽을 구현할 수 있는 저작도구를 선보였다. Unreal 5.0은 실제 사람처럼 보이는 디지털 휴먼 기술(Digital Human)

이나 라이팅 및 합성 기술의 도입으로 이전보다 더 사실적이고 더 빠르게 구현할 수 있다. 언리얼 엔진은 사용자들의 의견을 적극적으로 반영한 버전의 업데이트가 자주 이루어지며 최신 버전 또한 무료로 업데이트가 가능하다. 오픈소스 정책으로 사용자가 원하는 어떤 것이든 자유롭게 제작할 수 있어 게임이나, 영화, 인테리어, 가상현실(VR) 등에 다양하게 활용이 가능하며, 최근에는 영화 콘텐츠 제작을 위한 시네마틱 기술도 더욱 보강되고 있는 추세이다 [6].

[표 1]에서 보는 바와 같이 텍스처링과 애니메이팅을 언리얼의 블루프린트(Blueprint) [7]로 제작해야 한다는 번거로움이 있지만 블루프린트로 제작해놓으면 여러 번 중복하고 변환하여 사용할 수 있게 된다. 라이팅 역시 블루프린트로 규격화하여 비슷한 느낌의 라이팅을 한꺼번에 조정할 수 있다. 비주얼 스크립팅 언어인 블루프린트와 프로그래밍 스크립트 언어를 함께 사용하여 복잡한 구조나 퍼포먼스가 필요한 부분은 프로그래밍 언어로 구성하고, 가변성이 높은 부분은 디자이너가 간단히 변경할 수 있게 블루프린트로 구성하여 상호 보완적으로 활용 가능하다. 때문에 타 직군과의 협업이 원활하다.

영상편집 방식은 게임엔진 내에서 '시퀀서'로 개발되어있어 도구 안에서 한 번에 할 수 있다. 오프라인 파이프라인에서 렌더링을 모두 마친 후에 합성의 단계를 거쳐 영상편집을 하는 것과는 차이가 있다.

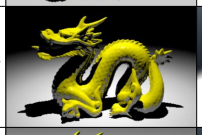
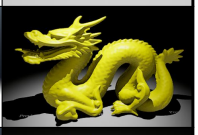
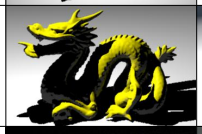
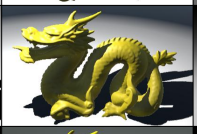
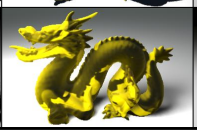
### 2.3 유니티와 언리얼 엔진의 라이팅 종류

렌더링은 라이팅과 밀접한 관계가 있다. 화면 안에 설치한 라이팅은 3D 에셋(asset)과의 연산 과정을 통해 렌더링 이미지로 시각화되며 작업물을 확인하기 위한 일정 시간이 소요된다. 반면 게임엔진에 설치한 라이팅은 이러한 시간 소요 없이, 실시간으로 최종 이미지를 확인하고 피드백을 바로 진행한다 [3].

오프라인 3D 프로그램 마야의 라이트 종류는 [표 2]와 같이 스팟(Spot) 라이트, 디렉셔널(Directional) 라이트, 에어리어(Area) 라이트, 포인트(Point) 라이트이고, 유니티의 라이트 종류는 스팟 라이트, 디렉셔널 라이트, 포인트 라이트, 에어리어 라이트, 메쉬(Mesh) 라이트로 각각의 라이트는 다른 속성을 가지고 있다. 언리얼 엔진의 라이트 유형은 다섯 가지, 디렉셔널, 포인트, 스팟, 스카이, 렉트(Rect) 라이트로 기본적인 속성은 마야나 유니티의 라이트와 유사하다. 디렉셔널 라이트는 야외 주광원 또는 무한히 먼 거리에서 빛을 쏘는 것처럼 보이는 광원에 주로 사용된다. 포인트 라이트는 고전적인 "전구"와 같은 라이트로, 한 광원을 중심으로 모든 방향으로 빛을 뿜는다. 스포트 라이트도 한 지점에서 빛을 뿜으나, 원뿔 형태로 빛이 제한된다. 스카이 라이트는 씬의 배경을 캡처하여 레벨의 메시에 라이팅으로 적용한다. 마야와 언리얼 엔진, 유니티의 공통된 라이트 속성은 빛의 강도, 컬러, 그림자의 부드러움 정도와 그림자의 노이즈를 조절하는 샘플(sample)을 조절하는 기능이 있다 [3]. 각 라이트는 화면의 크기와 위치, 시간, 종류와 분위기에 따라 다르게 세팅 되

는데, 라이트의 종류와 특성을 이해하고 프로젝트의 목적에 맞게 라이트의 수와 종류를 선택하는 것이 중요하다.

[표 2] 마야, 언리얼 엔진, 유니티의 라이트 종류
[Table 2] Light Types in Maya, Unreal Engine and Unity

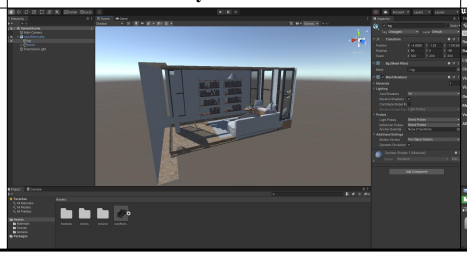
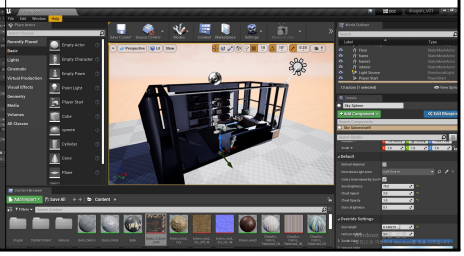
| 종류 | 특징 | Maya | Unity | Unreal Engine |
|---|---|---|---|---|
| Directional | ·태양광이나 월광처럼 무한대의 거리에서 오는 빛 |  |  |  |
| Spot | ·손전등처럼 방향성을 갖고 원뿔 형태로 일정 영역에 빛을 비추는 광원 |  |  |  |
| Point | ·텅스텐 전구처럼 광원을 중심으로 전 방향으로 발산하는 광원 |  |  |  |
| Area/Rect | ·직사각 형태의 빛, 부드러운 그림자<br>·텔레비전이나 모니터 화면, 벽처럼 직사각 영역을 비추는 광원 |  |  |  |

## 3. 유니티와 언리얼엔진의 렌더링 비교 실험

### 3.1 실험 환경

본 연구의 소프트웨어 및 하드웨어 실험 환경은 GPU geforce 3050, CPU Intel i7, Memory 32MB 환경에서 실내 모델링을 사용했다. 소프트웨어는 유니티 2020 3.21 버전을, Unreal Engine 4.27 버전을 사용했다.

[표 3] 유니티와 언리얼 엔진의 작업공간
[Table 3] Workspace of Unity and Unreal Engine

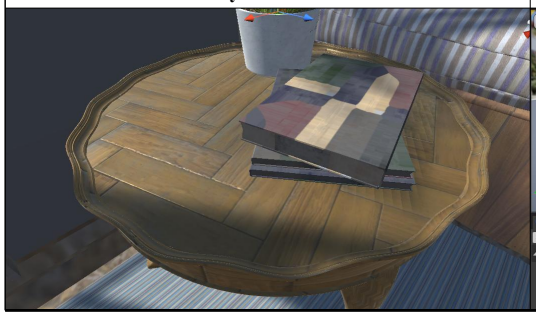
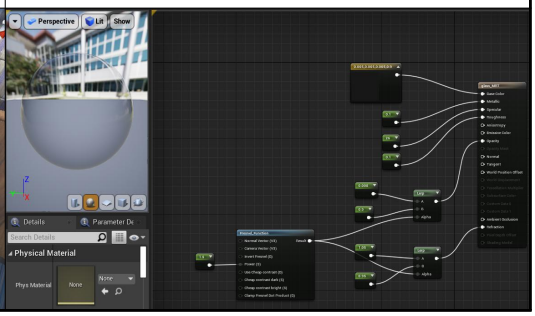
| Program | Unity 2020 3.21 | Unreal 4.27 |
|---|---|---|
| 작업공간 |  |  |

### 3.2 쉐이딩과 텍스춰링

3D 프로그램 마야에서 모델링된 데이터를 유니티와 언리얼로 불러와 씬 세팅을 시작했다. 유니티에서는 Fixed Function Shader, Vertex & Fragment Shader, Surface Shader 3가지 종류의 쉐이더가 있는데, Matrix같은 행렬연산이 자동으로 되는 Surface Shader를 적용했다. 다음 단계로 Surface Shader의 Smoothness, Metallic, Normal map을 적용해 오브젝트에 스펙큘러과 범프 등의 디테일을 더했다.

언리얼 엔진에서는 마야에서 불러온 오브젝트의 재질을 블루프린트로 재작업 했다. [표 4]의 유리 텍스춰처럼 블루프린트를 활용해 Shader의 Specular, Reflection, Roughness의 수치를 조절해 모델링에 리플렉션과 투명감 등의 디테일을 더했다. 블루프린트는 비주얼 스크립팅 시스템으로 언리얼 에디터에서 노드 기반 인터페이스를 사용해 게임 요소를 만드는 개념을 토대로 한다.

[표 4] 유니티와 언리얼 엔진의 쉐이더 적용
[Table 4] Shader Applied in Unity and Unreal Engine

| Unity 2020 3.21 | Unreal 4.27 |
|---|---|
|  |  |

### 3.3 라이팅과 렌더 세팅

#### 3.3.1 키 라이트(Key Light) 세팅

이번 실험의 시간대는 데이타임(Day Time)의 바다 배경으로 창문 사이로 강한 태양빛이 비치는 컨셉으로 설정했다. 키 라이트는 조명을 설치할 때 가장 먼저 설치해야 하는 것으로, 태양빛을 표현하기 위해 디렉셔널 라이트를 키 라이트로 설정했다 [표 5 참조]. 디렉셔널 라이트는 패러렐(Parallel) 라이트로 넓은 공간까지 뻗어가는 성질을 가지고 있어 태양빛이나 달빛을 구사하는데 주로 쓰인다. 라이트의 Intensity, 컬러 등을 조절해 낮 시간대 분위기를 만들어 주었다.

게임엔진에서는 움직임이 없는 정적인(static) 물체에 배치된 빛의 경로를 계산해 원본 텍스처에 추가로 베이크(bake)된 텍스처를 더해 화면에 빛을 표현하는 방식을 쓰는데, 이를 라이트맵핑

(Lightmapping) [5]이라고 한다. 라이트맵핑을 활용해 하드웨어에 더해지는 부담을 줄일 수 있다. 고정된 조명을 맵핑으로 출력해 실제 조명은 아니지만, 조명과 같은 효과를 줄 수 있다. 언리얼 엔진에서는 레벨에 지오메트리 또는 라이트를 이동 혹은 추가시킬 때마다 라이트맵을 리빌드(Re-Build)해야 정확한 표현이 가능하다.

테스트 결과 라이트의 베이크타임이 유니티와 언리얼에서 각각 5분과 8분 소요되었는데, 이는 테스트한 배경은 실제 프로덕션의 관점에서 보면 간단한 모델링으로, 폴리건 수가 많은 복잡한 구조의 에셋을 사용하면 베이크 타임이 크게 차이가 날 것으로 예상된다.

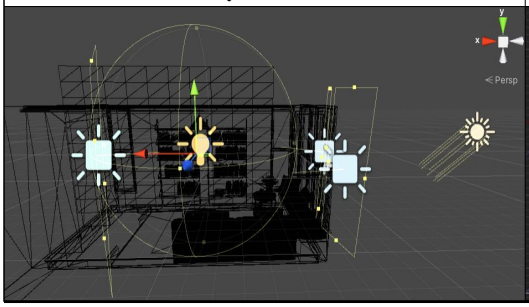
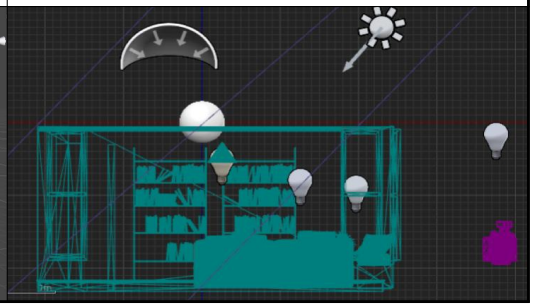
[표 5] 디렉셔널 라이트가 키 라이트로 적용된 실내 배경
[Table 5] Directional light applied in a room as key light

| Unity 2020 3.21 | Unreal 4.27 |
|---|---|
|  |  |

### 3.3.2 추가적인 라이트 세팅

먼저 키 라이트가 설치되면, 실내에 있는 광원 소스들을 파악하고 그에 맞게 추가적으로 필요한 라이트를 설치한다. 이번 테스트는 낮 시간대이므로 [표 6]과 같이 창 반대 벽 쪽에 약한 강도의 필 라이트를 창 옆쪽엔 렉트 라이트를 추가해 소파와 소파 위 쿠션의 가장 자리에 태양 빛을 더해주고, 천장 램프자리에 조명 소스들을 설치해 은은한 느낌의 광원들을 추가했다.

[표 6] 키 라이트와 필 라이트가 적용된 실내 배경의 와이어 프레임 모드
[Table 6] Wire Frame Mode, Applied Additional Lights in Unity and Unreal Engine

| Unity 2020 3.21 | Unreal 4.27 |
|---|---|
|  |  |

설치가 끝난 후, 빌드하여 최종 렌더 이미지를 생성했다. 조명 설정이 끝나면, 리플랙션(Refletion)을 세팅한다. 유니티에서는 쉐이더의 리플랙션을 정확하기 표현하기 위해 Reflect Probe를 활용하는데, 연산을 최소화하기 위해 Reflect Probe로 연산할 영역을 비어있는 공간 없이 꽉 채워 주었다. 언리얼에서는 리플렉션 캡쳐(refelection capture)를 활용하였는데, 스카이 라이트의 모빌리티(mobility)를 무버블(movable)로 바꾸어 주어 Reflection이 정확히 연산되도록 했다.

### 3.3.3 포스트 프로세싱(Post Processing)과 최종 렌더링

마지막 단계로, 포스트 프로세싱 [8]을 적용했다. 오프라인 렌더에서는 렌더링이 끝난 데이터를 누크(Nuke)와 같은 합성 프로그램에서 불러와 합성한 후 최종 결과물을 완성하게 되지만, 게임엔진에서는 포스트 프로세싱을 합성툴 대신 활용할 수 있다. 유니티와 언리얼 모두 포스트 프로세스 툴을 제공하고 있으며, 속성을 조정하여 화면 색조 및 블러링, 렌즈 플레어 등과 같은 효과를 추가하거나, 플레이어와의 상호작용 방식을 정의한다.

이번 실험에서는 [표 7]과 같이 Color Grading으로 차가운 색감을 더해주고 offset으로 어두운 책장 영역을 밝게 올려주어 형태감을 살리고 창가에서 오는 태양빛과 대치되도록 따뜻한 기운을 더했다. Bloom으로 Glow 이펙트를 주고 Vignetting으로 화면 가장자리를 어둡게 처리해 필름 효과를 더했다.

[표 7] 키 포스트 프로세스 적용 전, 적용 후

[Table 7] Before and After PostProcess

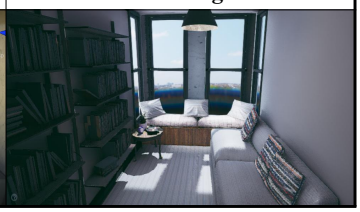
| Program | Unity 2020 3.21 | Unreal 4.27 |
|---|---|---|
| Postprocess적용 전 |  |  |
| Postprocess적용 후 |  |  |

## 3.4 라이팅·렌더링 실험 결과

3D 프로그램 마야와 언리얼 엔진, 유니티를 활용한 라이팅·렌더링 실험 결과, 세 프로그램이 모두 비슷한 라이팅 속성과 타입을 가지고 있어 비슷한 퀄러티의 결과를 만들어 낼 수 있었다 [표 8].

[표 8] 마야, 유니티와 언리얼 엔진의 최종 결과물 비교

[Table 8] Final Render Images Comparison

| Maya | Unity | Unreal Engine |
|---|---|---|
|  |  |  |

아래 [표 9]에 정리된 것처럼, 마야는 실내 배경에 6개의 라이트를 세팅, Directional light로 창에서 들어오는 태양빛을, 4개의 Area light를 창가, 천장과 벽의 Fill Light로, 1개 mesh 라이트를 전등에 설치했다. 마야의 기본 쉐이더인 블린(Blinn)을 사용해 스펙큘러를 더했다. 렌더 샘플은 노이즈가 적은 프로덕션 퀄러티로 렌더했을 때, 프레임당 렌더 시간은 29분 54초가 걸렸다. 라이트 세팅 작업 시간은 약 40분이 소요되었다.

[표 9] 라이팅과 렌더링 실험 결과

[Table 9] Comparison of Lighting and Rendering Results

| Item | | Maya | Unity | Unreal Engine |
|---|---|---|---|---|
| 라이트 종류 | | 1 Directional light, 4 Area lights, 1 Mesh light | 1 Directional light, 3 Area lights, 1 Point light | 1 Sky light, 1 Directional light, 4 Rect lights, 1 Point light |
| 총 사용된 라이트 수 | | 6 개 | 5 개 | 7 개 |
| 쉐이더 | | Blinn | Standard Shader | 블루 프린트 |
| 렌더 샘플 세팅 | | Camera(AA) 5, Diffuse 4, Specular 4, SSS 2, Transmission 2, Volume Indirect 2 | Lightmap Resolution 64, Direct Sample 128, Environment Sample 512, Indirect Sample 256 | Spatial Sample Count 8, Compression setting 100 |
| 작업 단계 | | 5 단계 | 4 단계 | 4 단계 |
| 작업 과정 별 작업 시간 | Layout | 10 분 | 10 분 | 10 분 |
| | Shading | 40 분 | 30 분 | 40 분 |
| | baking | 0 분 | 5 분 | 8 분 |
| | Lighting | 40 분 | 20 분 | 20 분 |
| | Rendering | 30 분 | 0 분 | 0 분 |
| | Composition | 30 분 | 0 분 | 0 분 |
| 총 작업 시간 | | 2시간 30분 | 1시간 5분 | 1 시간 18분 |

유니티에서는 실내 배경에 총 5개의 라이트를 세팅했다. Directional light로 창에서 들어오는 태양빛을, 3개의 Area light를 창가와 천장, 벽의 필 라이트로 조명했다. Point light는 전등에 설치해 은은한 실내조명을 완성했다. 라이트를 세팅한 작업시간은 약 20분이 소요되었다.

언리얼에서도 마찬가지로 Directional light로 창에서 들어오는 태양빛을, Sky light로 환경광을, 1개의 Point light, 4개의 Area light를 실내등과 창가 필 라이트로 설치해 총 7개의 라이트가 사용되었다. 라이팅 작업 시간은 약 20분이 걸렸다. 라이트를 설치할 때마다 테스트 렌더할 필요 없이 바로 확인할 수 있어 기존 오프라인 3D 프로그램보다 작업 시간이 절감되었다. 단지 라이트를 빌드하는데 8분 정도의 시간이 추가되었다. 실시간렌더이므로 렌더링 시간은 불필요했다.

게임엔진은 라이팅를 설치할 때마다 빌드해야 하는 번거로움이 있지만, 빌드 시간이 수분으로 렌더링을 기다리는 시간에 비한다면 작업 시간에 크게 영향을 주지는 않았다. 모델링 데이터가 크면 빌드 시간이 수 십분도 걸릴 수 있으나 한번 빌드를 하면 다른 카메라 각도에서 라이팅을 렌더 없이 확인할 수 있고, 기다림 없이 렌더되는 점이 장점으로 드러났다. 유니티와 언리얼은 합성 패스(pass)를 오프라인 렌더러와 같이 다양하게 렌더할 수 없다는 단점이 있지만, 게임엔진의 포스트 프로세스 기능을 대안으로 활용할 수 있었다.

작업 과정의 차이는 마야는 물리 기반 라이팅을 여러 번의 테스트 렌더를 거쳐 최종적으로 결과물을 만들 수 있었다. 또한 합성작업을 위한 라이팅 패스(pass)를 다양하게 렌더할 수 있어서 합성에 유리한 점이 게임엔진과의 차이점으로 드러났다. 이번 실험 결과 룩 개발(Look Develop.)과 직접적으로 관련이 있는 작업 과정인 쉐이딩, 라이팅, 렌더링에서 유니티와 언리얼이 더 유리한 것으로 확인되었다.

## 4. 결론

본 연구는 유니티와 언리얼 엔진, 두 게임엔진을 비교하여 3D 제작 파이프라인에 게임엔진의 라이팅·렌더링을 접목할 방법을 제시하고자 했다. 이를 위해 실내 모델링으로 마야와 유니티, 언리얼 엔진에서 렌더링을 비교 분석했다. 실험 결과 마야에서 제작한 배경을 쉽게 게임엔진으로 불러올 수 있고, 라이팅 속성 또한 두 엔진 모두 기존의 3D 프로그램과 유사한 속성을 가지고 있어 기존 오프라인 렌더링에 익숙한 사용자도 큰 어려움 없이 접근할 수 있었다.

폴리건의 수가 많은 모델링이나 복잡한 합성이 필요한 프로젝트는 3D 그래픽 프로그램으로 시퀀스(sequence) 렌더하여 합성하는 것이 높은 수준의 작품에는 유리할 것으로 파악된다. 그러나 TV 시리즈처럼 간단한 합성의 프로젝트는 유니티나 언리얼도 비슷한 수준의 결과를 만들어내는 것을 확인했다. 분석 결과, 두 게임엔진은 모두 오프라인 방식과 비교하여 렌더링 시간과 비용 절감이 가능하며 렌더팜이 필요한 정도의 대규모 렌더가 필요 없어 제작비용이 절감되는 것이 가장 큰 장점으로 부각 되었다.

게임엔진에는 장단점이 존재하고, 게임엔진을 선택하는 요소로 전문 지식, 팀워크, 리소스 및 시간과 같이 다양한 변수가 있어 단적으로 특정 게임엔진이 좋다고 말할 수는 없다 [9]. 그렇지만 본

연구에서 두 게임엔진의 라이팅·렌더링 방식을 소개 및 비교함으로써 사용자에게 어떤 게임엔진이 특정 프로젝트에 더 적합한지에 대한 정보를 줄 수 있을 것이다. 이번 실험을 통해 게임엔진의 라이팅 세팅 방식, 유니티와 언리얼의 렌더 타임, 렌더 방식, 노이즈 정도 등을 분석해 봄으로써, 오프라인 렌더링 파이프라인에 익숙한 사용자들에게 실시간 렌더엔진을 활용할 수 있는 가이드를 제공했다는 것에 의의를 둔다. 향후 연구 계획은 게임엔진을 활용해 애니메이션 영상을 제작해 기존 오프라인의 파이프라인과의 작업과정을 비교 분석할 필요가 있을 것이다.

## References

[1] J. P. Jung, Unity Shader Strt up, VIEL Books, 2021.

[2] 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, Xiangtan, China, pp. 236-239, doi: 10.1109/ICICTA49267.2019.00057.

[3] G. Y. Kim, “Comparative Analysis of Real-Time Rendering Methods”, 2021 NCISS Winter Conference, December 29-30, 2021, Jeju, Korea, pp. 77-80.

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

[5] M. K. Kim, the Game Graphics, Viel Books, 2015.

[6] T. Shannon, Unreal Engine 4 for Design Visualization, Acorn Publish, 2018.

[7] B. B. Ramos, J. Doran, K. Y. Kim, Unreal Engine 4 material, Acorn Public, 2020.

[8] H. I. Cho, S. J. Shin, “A Study on the Characteristics of Game Engines for Next-Generation Multimedia Production Education (Focusing on Unity and Unreal Game Engine)”, Korean Society For Computer Game, vol. 34, no. 3, September 2021, pp. 29-38.

[9] A. Šmíd, “Comparison of Unity and Unreal Engine”, Bachelor thesis, Faculty of Electrical Engineering Department of Computer Graphics and Interaction, Czech Technical University, Czech Republic, 2017.