JKSCI

한국컴퓨터정보학회논문지
Journal of The Korea Society of Computer and Information
Vol. 27 No. 1, pp. 23-32, January 2022
https://doi.org/10.9708/jksci.2022.27.01.023

Comparative Analysis of Three-Dimensional Real-Time Rendering Methods

Gum-Young Kim*, Byong-Kwon Lee*

*Professor, Dept. of Multimedia, Seowon University, Chungbuk, Korea
*Professor, Dept. of Multimedia, Seowon University, Chungbuk, Korea

[Abstract]

Image production using three-dimensional (3D) programs undergoes a process called rendering to visualize 3D data. Because this process is time-consuming and costly, the reduction of rendering cost has emerged as an important problem that requires resolution. 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 Unity were compared and analyzed. The analysis results indicate that Unity 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.

▸ **Key words**: Lighting and Rendering, Real-Time Rendering, Game Engine, 3D Production Pipeline, Rendering Methods

[요 약]

3D 프로그램을 활용한 방식의 영상 제작과정은 3D 데이터를 시각화하는 렌더링이라는 과정을 거친다. 이 과정은 제작 시간과 작업비용의 부담이 커서 제작자로서는 렌더 비용을 줄이는 것이 중요한 이슈로 대두되고 있다. 본 연구는 실시간렌더엔진인 게임엔진의 렌더링을 영상 제작에 접목해 렌더 시간을 줄여 기존의 제작 시간을 단축하는 방법을 제시하고자 한다. 본 연구의 렌더링 실험을 위해 로봇과 실내 모델링을 3D 제작 프로그램인 마야(Maya)와 게임엔진인 유니티(Unity)로 렌더링을 비교 분석했다. 분석 결과, 유니티는 실시간으로 렌더 되어 렌더 비용을 줄일 수 있고, 렌더된 이미지의 퀄러티 또한 마야에서 렌더된 이미지와 유사한 결과를 나타냈다. 이번 실험을 통해 유니티의 렌더링을 마야와 비교해 분석해 봄으로써 기존의 제작 시간을 단축하는 방법을 제시하고 사용자가 실시간렌더엔진에 접근할 수 있는 가이드를 제공한다.

▸ **주제어**: 라이팅과 렌더링, 실시간렌더링, 게임엔진, 3D 제작 파이프라인, 렌더링 기법

• First Author: Gum-Young Kim, Corresponding Author: Byong-Kwon Lee
*Gum-Young Kim (kimky_j@hotmail.com), Dept. of Multimedia, Seowon University
*Byong-Kwon Lee (sonic747@daum.net), Dept. of Multimedia, Seowon University
• Received: 2021. 11. 12, Revised: 2021. 12. 17, Accepted: 2022. 01. 05.

http://www.ksci.re.kr pISSN:1598-849X | eISSN:2383-9945

I. Introduction

3D애니메이션이나 영화의 VFX와 같이 3D 프로그램을 활용한 방식의 영상제작 과정은 3D 데이터를 시각화하는 렌더링(Rendering)을 한다. 렌더링에는 많은 시간이 소요되어 대부분의 애니메이션 회사나 영화 프로덕션에서는 3D데이터를 시퀀스(Sequence) 이미지로 렌더하기 위해서 렌더팜(Render Farm)이라는 고가의 렌더링 서버[1]를 구비하거나 렌트하고 있다. 또한 영상기술의 발달과 4K 해상도의 고화질 TV 및 2K 해상도를 지원하는 고화질 모바일 디바이스의 보급으로 영상을 제작하는 제작자 입장에서는 렌더링 타임에 대한 부담은 갈수록 가중되고 있다[2].

이러한 근본적인 렌더링 타임을 해결하기 위해 최근에는 유니티(Unity)나 언리얼엔진(Unreal Engine)과 같은 실시간렌더엔진인 게임엔진을 애니메이션이나 영화, 드라마, 건축과 같은 비게임 분야에 접목[3]하려는 시도가 국내외 주요 스튜디오를 중심으로 이루어지고 있다. 게임 개발 초창기의 게임엔진은 표 1과 같이 둠(DOOM) 시리즈 그래픽 정도의 퍼포먼스로 당시에는 혁신적이었지만 제대로 된 영상을 만들 수 있는 수준이 아니었다[4]. 이후 컴퓨터 하드웨어의 발전으로 2016년에 출시된 언차티드4는 영상과 게임 플레이 화면을 구분할 수 없을 정도로 발전하게 되었다.

Table 1. Graphic Development in Game

Company	id Software	NAUGHTY DOG
Game Name	Doom	Uncharted
Service Year	1993	2021
Image		

3D영상의 렌더링은 라이팅과 밀접한 관련이 있다. 화면 구성 내에 설치한 라이팅은 3D에셋과의 연산 과정을 통해 렌더링 이미지로 추출되며 작업물을 확인하기 위한 일정한 시간이 소요된다. 반면 게임엔진에 설치한 라이팅은 실시간으로 최종 이미지를 확인하고 수정이 있을 시 피드백을 바로 진행한다.

본 연구는 실시간 게임엔진을 이용하여 3D영상을 제작해 봄으로써 3D영상 제작 파이프라인이 가지고 있는 한계점을 개선하고 실시간 렌더엔진인 게임엔진의 라이팅·렌더링을 영상 제작에 접목해 렌더 시간을 줄여 기존의 제작 시간을 단축하는 방법을 제시한다.

본 연구의 구성은 2절에서는 실시간렌더엔진인 유니티의 라이팅과 렌더링, 3D프로그램 마야(Maya)의 라이팅과 렌더링, 그리고 라이팅의 종류와 글로벌일루미네이션(Global Illumination, 이하 GI) 등의 라이팅 기법 관련연구를 진행하고, 3절에서는 로봇 모델링을 마야와 유니티로 렌더링을 비교 분석했다. 4절에서는 3절에서 분석된 방법으로 유니티를 이용해 실내 모델링으로 렌더링을 테스트해 작업시간과 렌더 시간을 마야에 비해 얼마나 줄일 수 있는지, 결과물의 노이즈 정도, 마야와 유사한 렌더링 퀄러티를 도출할 수 있는지를 확인하고, 5절에서는 연구의 결론 및 향후 계획을 제시한다.

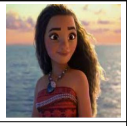
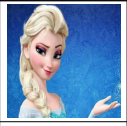
II. Preliminaries

1. 3D Modeling Program "Maya"

AutoDesk에서 만든 3D 프로그램으로 세계 영화 특수효과 업계의 산업표준 그래픽 소프트웨어이다. 모아나, 겨울왕국, 몬스터 주식회사, 매트릭스, 스파이더맨, 아바타 등 국내외 대다수의 영상 프로젝트에 사용되고 있으며, 모델링에서 애니메이션, 시뮬레이션, 라이팅, 렌더링까지 합성을 제외한 영상 제작과정에 필요한 모든 기능을 담고 있다. 특히 마야는 모델링과 애니메이션에 특화되어 있는데, 넙스(nurbs) 방식의 모델링과 폴리(poly) 방식의 모델링이 둘 다 가능하다는 장점이 있다. 폴리건 모델링 방식은 정점, 모서리 및 면을 기반으로 하는 형상을 사용하여 3D 모델을 제작하고, 넙스 방식은 기하학적 원형 및 그려진 곡선에서 3D 모델을 구성할 수 있다[5].

마야는 영상제작에 관련한 광범위한 기능을 구현하고 있는데, 그 중 마야에 기본적으로 장착된 렌더러는 마야소프트웨어(maya software)와 아놀드(arnold) 렌더러가 있다. 마야소프트웨어 렌더러는 글로벌일루미네이션(본 논문 2절 참조)을 지원하지 않고, 조절할 수 있는 속성이 한정적인 렌더러로 렌더 속도는 아놀드에 비해 빠른 편이다. 아놀드 렌더러는 글로벌일루미네이션, 레이 트레이싱(Ray-tracing), 반사, 굴절 등의 고난위도 렌더링을 정확하게 구사한다. 단점은 렌더타임이 오래 걸린다.

Table 2. Movies Made using Maya

Image				
Title	Avata	Moana	Frozen	Spiderman
Release	2009	2016	2019	2019

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[6].

3D영상 파이프라인 중 프리 프로덕션(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. 포스트프로덕션(Post-Production)단계는 렌더링 과정을 통해 각 프레임마다 이미지를 추출하는 과정과 영상을 편집하는 과정으로 구성된다[7].

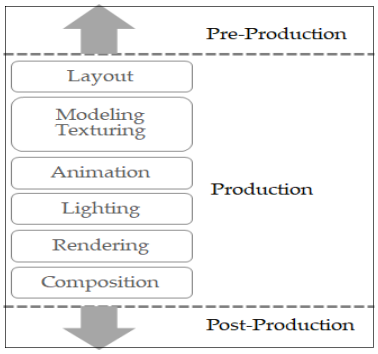


Fig. 1. 3D Animation Production Pipeline

2. Real-time Render Engine Comparison

가상현실, 증강현실, 인터렉션 미디어 아트, 디지털 트윈과 같이 게임이 아닌 다양한 분야에서 게임엔진 사용이 급증하고 있다. 가상현실, IoT, Big Data 등의 신기술이 결합된 영역인 디지털 트윈은 단어에서처럼 현실 물리세계의 오브젝트와 가상세계의 모델이 서로 유사하게 존재한다는 의미에서 '쌍둥이(Twin)'이라고 불리며, 두 세계의 존재를 지칭하고 있다. 게임 엔진을 이용하면 저비용으로 단시간에 디지털 트윈을 구축할 수 있는 장점이 있다[8].

게임엔진은 애니메이션, 디자인 또는 그래픽과 같은 시각화를 즉시 생성하고, 렌더링 시간을 기다리지 않고 디자인을 실시간으로 수정 및 결과물을 즉시 확인할 수 있어 실시간렌더엔진[9]이라고도 불린다. 유니티 엔진(Unity 3D)과 언리얼엔진(Unreal)은 대표적인 게임엔진으로 여러 분야에서 저작도구로 사용되고 있는 추세이다.

2.1 Unity Game Engine

유니티는 2004년 8월 Unity Technologies가 개발한 게임엔진으로 게임 개발 시장의 62% 이상을 차지하는 글로벌 콘텐츠 제작도구이다. 국내에 20만명이 넘는 사용자가 유니티를 활용해서 콘텐츠를 제작하고 있다. 게임엔진으로 널리 알려졌지만, 현재 영화, 애니메이션, 광고, 건축 등 전 산업에 걸쳐 활용되고 있다. 2018년 유니티는 Siggraph 2018에서 영화를 위한 유니티 엔진의 활용에 대해 발표하였고, 제작 분야별로 최적의 파일을 설정해 주는 초기 단계를 구성하고, 렌더링과 쉐이더 기술을 발전시켜 고품격의 그래픽 구현하는 데 주력하고 있다[10].

유니티는 복잡한 3D게임을 간단하게 만들 수 있도록 해주는 그래픽 환경의 통합형게임 개발 프로그램으로 게임엔진과 에디터가 하나로 구성된다. 유니티에서 개발된 게임 소스는 iOS, 안드로이드, 윈도우, 블랙베리, 리눅스, 웹브라우저, 플래시, 플레이 스테이션, X박스 360, 윈도우폰, Wii U 등 수 많은 기기로 배포될 수 있다. 유니티는 간단하고 직관적인 인터페이스의 최적의 작업 환경을 제공하고 게임을 개발할 수 있는 강력한 도구모음을 가지고 있다. 또한 C#, JavaScript 등의 프로그래밍 언어를 적용하여 게임을 구동시킬 수 있다. 3D 모델링 삽입, 조명, 그림자, 특수효과, 오디오, 물체의 질감, 지형, 물리법칙, 인공지능 등을 사실적으로 표현할 수 있다[11].

2.2 Unreal Engine

언리얼 엔진(Unreal Engine)은 미국의 에픽 게임즈에서 개발한 3차원 게임엔진이다. 1994년부터 현재까지 꾸준한 개발을 통해 발전하고 있으며, 2021년 Unreal 5.0을 발표하고 루멘, 나나이트와 같은 기술을 투입해 방대한 양의 메쉬(mesh)를 계산하여 이미지를 구현할 수 있는 저작도구를 선보였다. 최근 발표된 새로운 버전 5.0은 실제 사람 같이 보이는 기술(Digital Human)이나 라이팅과 합성기술의 새로운 도입으로 이전보다 더 사실적이면서 더 빠르게 구현할 수 있게 되었다.

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[10].

Table 3. Unity and Unreal Comparison

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

2.3 Real-time Render Engine Pipeline

게임엔진 기반의 3D영상 제작공정(Pipeline)은 프리프로덕션단계는 기존 파이프라인과 동일하나 메인프로덕션 단계에서 차이를 보인다. 3D 그래픽 툴로 제작된 캐릭터와 배경 오브젝트 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. 카메라 애니메이팅의 연출단계를 거친 후, 실시간으로 재생되는 장면을 추출하고 편집한다. 작업의 프로세스를 전통적인 스토리보딩 프로세스를 건너뛰고 게임엔진을 사용하게 되면 처음부터 컴포지팅(Compositing) 작업이 진행되어 렌더팜을 사용하지 않고 완성이 가능해진다. 이처럼 3D영상 제작공정에서는 일정 시간을 소모하는 렌더링 과정을 통해 결과물을 수시로 확인해야 하지만, 게임엔진 기반 영상작업은 실시간으로 제작된 영상을 바로 확인할 수 있어 피드백 반영시간을 절약할 수 있다는 것이 효율성 측면에서 큰 장점이다[7].

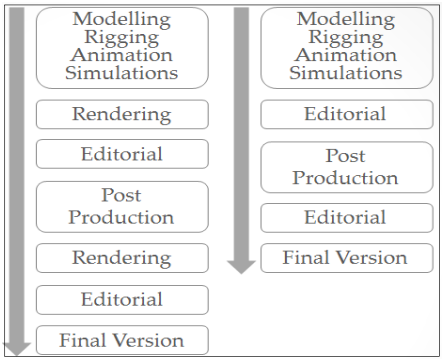


Fig. 2. Artist Workflows with Maya(left) and Unity(right)

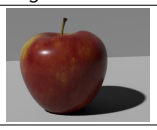
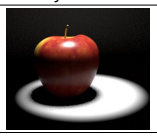
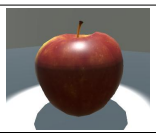
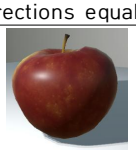
3. Lighting and Rendering Methods

3.1 Light Types

본 연구에서는 라이팅 기법을 마야와 유니티에서 비교 분석했다. 마야의 라이트 종류는 디렉셔널(Directional) 라이트, 스팟(Spot) 라이트, 포인트(Point) 라이트, 에어리어(Area) 라이트, 메쉬(Mesh) 라이트로 각기 다른 속성을 가지고 있다[표 4 참조]. 유니티의 라이트 종류는 디렉셔널 라이트, 스팟 라이트, 포인트 라이트, 에어리어 라이트, 메쉬 라이트로 기본적인 속성은 마야의 라이트와 유사하다. 마야와 유니티의 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 연구에서는 마야와 유니티에 공통으로 있는 디렉셔널 라이트, 스팟 라이트, 포인트 라이트, 에어리어 라이트, 메쉬 라이트, 5가지 라이팅을 유니티와 마야에서 비교했다.

Table 4. Light Types in Maya and Unity

Type	Maya	Unity
Direct-ional	·Parallel ·Simulate a very distant point light source	·Located infinitely far away and emits light in one direction only
		
Spot	·A beam of light evenly within a narrow range of directions that are defined by a cone	·Located at a point in the scene and emits light in a cone shape
		
Point	·Shines evenly in all directions	·Located at a point in the scene and emits light in all directions equally
		
Area	·Rectangular shape ·Resizable ·Longer to render	·Defined by a rectangle in the scene,
		
Mesh	·Mesh emits like a light itself	·Material option emitter
		

3.2 Direct Light vs Global Illumination (GI)

광원 소스가 태양 빛 하나로 출발할지라도 이 태양 빛이 하늘에서 산란하기도 하고 바닥에 부딪힌 뒤 튕겨 나오기도 한다. 물체에 부딪힌 빛이 바닥이나 벽에 다시 튕겨서 들어오기도 한다. 이렇듯 빛은 한 방향에서만 들어오는 것이 아니라 사방에서 들어와 오브젝트는 여러 방향에서 빛의 영향을 받게 된다. 아래 그림 3의 왼쪽 이미지와 같이 키 조명을 배치했을 경우가 기본적인 조명의 효과이다. 여기에서 더욱 사실적인 느낌을 주기 위해서는 GI와 같은 효과가 필요한데 GI는 기존의 방식에서 레이 트리이싱(Ray Tracing)으로 발전하여 두 표면 사이에 빛의 상호 반사를 구현하는 렌더링 알고리즘[12]을 총칭한다.

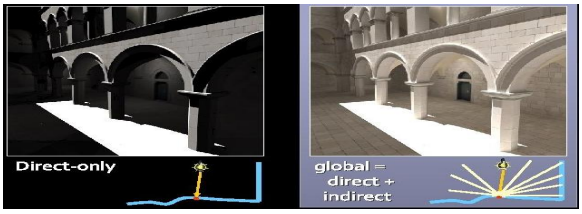


Fig. 3. Global Illumination Across Industries Course SIGGRAPH 2010 Jaroslav

GI를 모방하기 위해 유니티는 스카이박스(Skybox)를 이용한 환경 라이팅 방식을 제공한다. 새로운 씬(Scene)을 생성하면 기본적으로 Ambient Source가 Skybox로 되어 있어 GI가 자동 생성되고 Directional light 소스 하나만으로도 부드러운 라이팅이 적용된다.

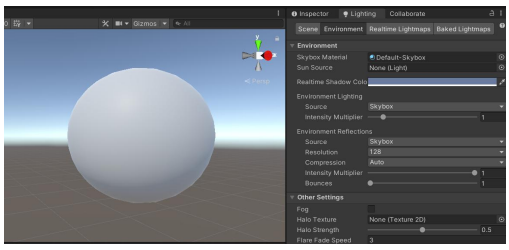


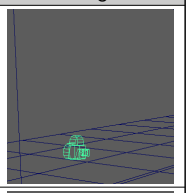
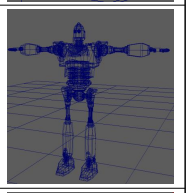
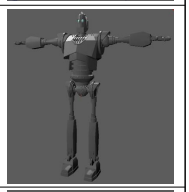
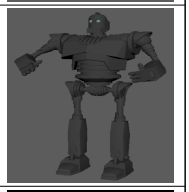
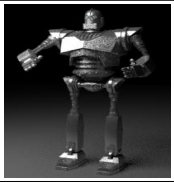
Fig. 4. Unity Global Illumination

III. The Proposed Scheme

1. Production Process in Maya

3D 애니메이션, 영화의 특수효과 등에 널리 쓰이고 있는 프로그램 마야의 렌더링을 유니티와 비교 분석했다. 마야의 프로덕션 작업과정은 표 5과 같이 레이아웃, 모델링, 쉐이딩·텍스춰링, 애니메이션, 라이팅, 렌더링, 합성으로 총 7단계로 구성된다. 본 연구에서는 로봇 모델링을 활용하여 레이아웃부터 합성까지의 과정을 마야에서 아래와 같은 세팅으로 진행했다.

Table 5. Production Process in Maya

Step	explanation	image
Step 1 Layout	·First step in production ·Decide what will be on the screen based on the storyboard ·Deal with complex camera movements	
Step 2 Modeling	·Modelled with Polygon for hard surface	
Step 3 Shading Texturing	·Defines glossiness, textures, etc on a surface ·Applied aiStandardSurface as we will render the image with Arnold renderer in Maya	
Step 4 Animation	·Key frame animation ·Setup bones and controllers to animate character	
Step 5 Lighting	·Setup 3 area lights for key light, fill light, rim light	
Step 6 Rendering	·Converted 3D data to images ·Used arnold renderer as it is a full Ray Traced / Global Illumination solution	
Step 7 Composition	·Combined all image sources and lighting passes to make final images ·Used Nuke program for composition	

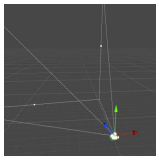
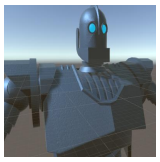
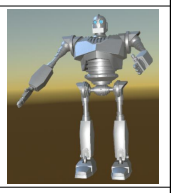
2. Production Process in Unity

유니티는 모델링, 애니메이션, 합성 기능은 제공하고 있지 않기 때문에 1단계 레이아웃에서 바로 마야에서 작업된 모델링 데이터를 유니티로 임포트(import)해 쉐이딩, 텍스춰 단계부터 시작했다. 유니티와 같은 물리엔진에서 실시간 렌더링을 구현하기 위해서는 엔진에 최적화하는 작업이 필

요하다. 그래서 모델링 데이터의 폴리건 수를 줄여 게임엔진에서 부하가 걸리지 않게 최적화해 유니티로 임포트했다.

쉐이딩은 마야만큼 다양하게 제공하고 있지 않으나, 머터리얼(Material)의 컬러, 메탈리스, 노말맵, 투명도 조절 등 기본적인 기능을 가지고 로봇에 필요한 쉐이딩을 만들었다. 애니메이션 또한 마야에서 작업된 데이터를 임포트하여 라이팅을 작업했다. 두 프로그램 모두 비슷한 라이팅 타입과 속성을 가지고 있어 근접한 퀄러티의 결과물을 만들어냈다.

Table 6. Production Process in Unity

Step	explanation	image
Step 1 Layout	·First step in production ·Decide what will be on the screen based on the storyboard ·Deal with complex camera movements	
Step 2 Import modeling	·Imported modelling data from Maya	X
Step 3 Shading Texturing	·Fixed Function Shader, Vertex & Fragment Shader, Surface Shader ·Applied Standard Shader as it does not require scripting and it is artist friendly.	
Step 4 Lighting	·Setup 3 area lights for key light, fill light, rim light	
Step 5 Baking	·Bake lights	X
Step 6 Rendering	·Not Applicable	X
Step 7 Composition	·Not Applicable	X

3. Comparison of Lighting and Rendering

작업 프로세스상의 차이는 마야의 경우 라이팅맵을 별도로 베이크할 필요가 없이 물리 기반의 라이팅을 테스트 렌더를 거쳐 최종 결과물로 만들 수 있고, 또한 합성을 위한 라이팅 패스들을 뽑을 수 있어 합성에 유리한 점이 유니티와의 차이점으로 드러났다.

이와 달리 유니티는 라이팅맵을 별도로 뽑아야 하는 번거로움이 있으나, 베이크 시간이 수분으로 작업시간에 크게 영향을 주지 않았다. 모델링 데이터가 클 경우는 베이크 시간이 수십분도 걸릴 수가 있으나, 결정적으로 한번 베이크를 하면 여러 카메라 각도에서 라이팅을 확인할 수 있고, 렌더 시간이 따로 필요 없어 실시간으로 렌더되는 점이 큰 장점으로 드러났다. 유니티는 합성 패스(pass)를 뽑을 수 없다는 단점이 있지만, 유니티의 포스트 프로세스 플러그인을 대안으로 활용할 수 있다.

Table 7. Comparison of Production Process in Maya and Unity

Item		Maya	Unity
No. of Steps		5 Steps	4 Steps
Working Time in stages	Layout	10 min	10 min
	Shading Texturing	1 hour	40 min
	baking	0 min	5 min
	Lighting	30 min	20 min
	Rendering	35 min	0 min
	Composition	30 min	0 min
Total Working Time		2 hours 45 min	1 hours 15 min

표 7을 보면 작업 시간에 있어 유니티와 마야의 가장 큰 차이점은 렌더링과 합성으로 실시간렌더러인 유니티는 렌더링 시간이 필요 없고, 다만 라이팅을 텍스춰에 입히는 베이킹 시간만 수분 필요했다. 라이팅과 쉐이딩을 세팅하는 작업 시간 또한 유니티가 각각 10분과 20분 더 짧게 소요되었는데, 그 이유는 마야에서는 테스트 렌더를 거쳐 쉐이딩, 라이팅 결과 확인이 가능한데, 유니티는 바로 확인이 가능해 작업 시간이 더 단축되었다. 아래 그림 5의 차트를 보면 그 차이가 확연히 드러난다. 쉐이딩, 렌더링, 합성에서 마야의 작업 시간이 더 많고, 베이킹 시간은 유니티가 5분 더 소요되었다. 이번 실험으로 룩 개발(Look Development)과 직접 관련이 있는 작업 프로세스들, 쉐이딩, 라이팅, 렌더링, 합성에서 유니티가 더 유리한 것으로 확인되었다.

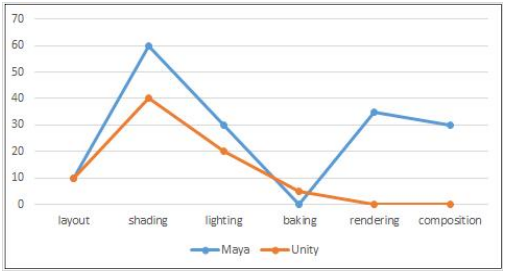


Fig. 5. Comparison of Working Time in Maya and Unity (units: min)

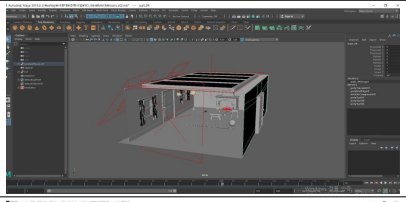
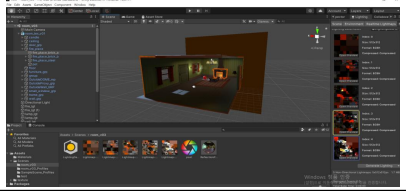
IV. Experimental Results

1. Experimental environment

본 연구의 하드웨어 및 소프트웨어 실험 환경은 CPU Intel i7, GPU geforce 3050, Memory 32MB 환경에서

실내 배경 모델링을 사용했다. 소프트웨어는 3D프로그램 마야 2019와 마야에 기본으로 설치된 렌더러 Arnold 3.2.2 버전을, Unity 2020.3.21.f1버전을 사용했다.

Table 8. Workspace of Maya and Unity

Program	workspace
maya 2019- Arnold Renderer	
unity 2020.3.21	

2. Shading and Texturing

유니티에서는 Fixed Function Shader, Vertex & Fragment Shader, Surface Shader 3가지 종류의 쉐이더가 있는데, Matrix 같은 행렬 연산이 자동으로 되는 Surface Shader를 적용했다. 기본적으로 마야에서 작업한 텍스처를 유니티의 쉐이더 레가시(Legacy)모드로 불러와 모든 오브젝트에 한 번에 텍스처 적용이 가능했다.

다음 단계로 그림 6과 같이 Surface Shader의 Metallic, Smoothness, Normal map을 적용해 모델링에 스펙과 범프 등의 디테일을 더했다.

Fig. 6. Surface shader applied in Unity

3. Lighting and Rendering

3.1 Scene Setting

라이팅을 설치하기 이전에 Environment Lighting을 세팅하게 되는데, 이번 테스트의 시간대는 저녁으로 달빛을 받아 창문 사이로 은은하게 달빛이 비치는 컨셉으로 설정했다. 이를 위해 Skybox Material은 Default로 설치된 데이터임(Daytime) 스카이를 None으로, Ambient Color를 어두운 파란 계열의 색으로 세팅해 밤 분위기를 조성했다.

3.2 Key Light Setting and Baking

조명을 설치할 때 가장 먼저 주요하게 설치해야 하는 것이 키 라이트이다. 우선 달빛을 구사하기 위해 Directional Light를 키 라이트로 설정했다[그림7]. Directional Light는 패러렐(Parallel) 라이트로 아주 넓은 공간까지 뻗어가는 평행 성질을 가진 라이트로 달빛이나 태양빛을 구사하는데 주로 쓰인다. Directional Light의 Intensity, 컬러 등을 조절해 밤 배경을 우선 만들어 주었다.

유니티에서는 움직임이 없는 정적인 오브젝트에 배치되어진 빛의 경로를 계산하여 원본 텍스처에 추가로 베이크된 텍스처를 더하여 화면에 빛을 표현하는 방식을 사용하는데, 이를 라이트맵핑(Lightmapping)[11]이라고 한다. 라이트맵핑을 활용하면 하드웨어에 가중되는 부담을 줄일 수 있다. 라이트맵핑은 고정된 조명을 맵핑으로 출력함으로써 실제 조명은 아니지만 조명과 같은 효과를 줄 수 있다. 라이트맵을 최종적으로 빌드(Build)하는 과정을 보통 베이크(Bake)라고 표현한다. Non-directional mode는 가장 기본적인 라이트맵 베이크 방식으로, 오브젝트를 기반으로 베이크한 라이트맵 텍스처에 입혀서 조명 연산에 대한 부하를 줄여주는 방식이다. Direional mode는 라이트맵 생성 시 한 장의 라이트맵과 한 장의 노멀에 대한 라이트맵이 생성된다. 즉, 빛을 받는 오브젝트에 대한 노멀 맵의 그림자까지도 표현하기 때문에 더 사실적이고 풍부한 라이트맵을 만들 수 있는 기법이다. 테스트 결과 Non-directional mode에서 baketime이 1분 25초, Directional mode에서는 baketime이 1분 49초로 렌더 노이즈 퀄러티는 크게 차이나지 않고 baketime은 24초 더 소요되었다. 이는 테스트한 모델링 배경은 노멀맵이 부분적으로 적용되어 있어 Directional mode에서 크게 효과를 보지 못한 것으로 분석된다. 노멀맵이 많이 적용된 에셋은 베이크 타임의 차이가 클 것으로 파악된다. 테스트한 배경은 프로덕션의 관점에서 보면 비교적 간단한 모델링으로, 실제 프로덕션에 적용되는 폴리건 수가 많은 복잡한 구조의 모델링을 사용하게 되면 Directioal Mode를 어떤 것으로 선택하느냐에 따라 베이크 타임이 크게 차이 날 것으로 예상된다,

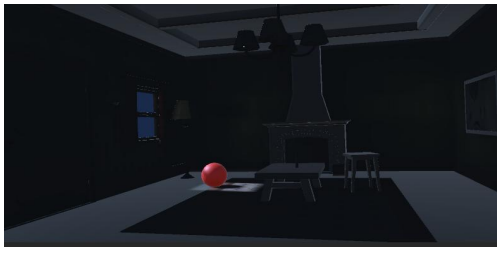


Fig. 7. Only Key light Applied in Unity

3.3 Additional Lights Setting and Baking

키 라이트가 설정되면, 실내에 있는 조명 소스들을 파악하고 그에 맞추어 부가적인 라이트들을 설치하게 된다. 이번 테스트는 밤 시간대이므로 그림 8과 같이 벽난로나 촛불, 램프와 같은 조명 소스들을 활용해 은은한 느낌의 광원들을 설치해 주었다. 설치가 끝난 후, 베이크(Bake)하여 라이트맵을 생성했다.

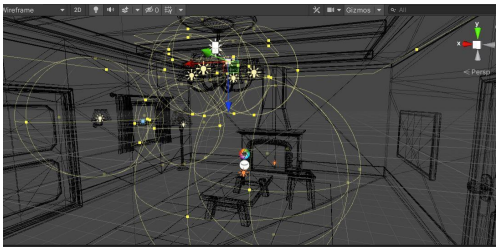


Fig. 8. Additional Lights in Unity Wire Frame Mode

조명 설정이 끝나면, 다음으로 리플랙션(Refletion)을 설정한다. 유니티에서는 쉐이더의 리플랙션을 제대로 받기 위해, Reflect Probe를 활용하게 되는데, 이때 주의할 점은 연산을 최소화하기 위해 그림 9와 같이 Reflect Probe가 연산할 영역을 빈 공간 없이 꽉 채워 주는 것이다.

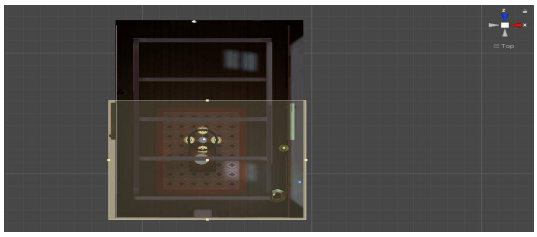


Fig. 9. Reflection Probe Area in Unity

4. Post Processing and Final Render

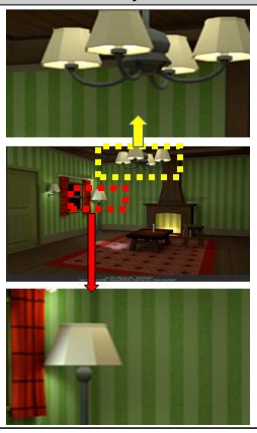
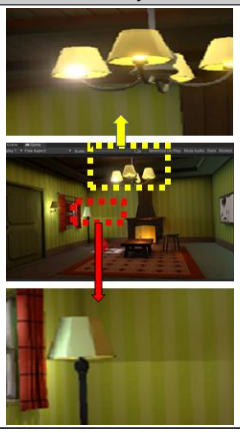
라이트 맵핑을 출력하게 되면 맵핑에 고정되기 때문에 움직이는 오브젝트나 캐릭터에는 영향을 줄 수 없다. 씬에서 움직이는 캐릭터나 오브젝트에 조명 효과를 주는 방법을 아래와 같이 제안한다.

우선 키 라이트로 사용된 Directional Light의 Mode는 Mixed로 설정하여, 정적이거나 동적인 오브젝트 모두에 적용이 되도록 설정한다. 동적인 배경 모델링 데이터는 Static으로 설정했다. Directional Light의 그림자가 실내에 제대로 연산 되지 않는 현상이 확인되었다. 다시 마야에서 벽체만 따로 모델링 하고 유니티에 FBX파일로 임포트해 방 안에 그림자가 캐스팅(Casting)되도록 설정했다. 그림 10 처럼 애니메이션 된 볼(ball)을 불러들여 씬 안에서 그림자가 제대로 작동되는 것과 라이트의 영향을 받고 있는 것을 확인했다.

Fig. 10. Animated Ball

마지막 단계로, 유니티에서 포스트 프로세싱(Post Processing)을 적용했다. 마야에서는 렌더링 후 합성작업을 거쳐 최종 결과물을 완성하게 되지만, 유니티에서는 합성작업을 대체할 만한 플러그인으로 포스트 프로세싱을 활용할 수 있다. 포스트 프로세싱은 기존에 렌더링 된 씬에 렌더링 효과를 더하는 작업이다[8]. 이번 실험에서는 표 9와 같이 포스트 프로세싱 옵션 중 Color Grading으로 따뜻한 색감을 더하고, Bloom으로 Glow 이펙트를 주었다. Mesh Light를 적용한 오브젝트에 Bloom효과가 생겼다. 그리고 Vignetting으로 화면 가장자리를 살짝 어둡게 눌러주어 필름 효과를, Ambient Occlusion에 옅은 오렌지색을 더해 벽난로나 조명에서 오는 따스한 기운을 면이 닿는 부분에 적용했다.

Table 9. Final Render Image Comparison

Maya	Unity
	

라이트 세팅이 끝나게 되면 마야에서는 이미지를 exr포맷으로 렌더팜에서 시퀀스 렌더해 Nuke나 After effect 같은 전문 합성 프로그램에서 합성하는 과정이 필요하다.

이에 비해 유니티에서는 타임라인을 꺼내서 마야에서 작업된 애니메이션 데이터를 오브젝트에 적용한다. 유니티

는 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.

5. Lighting and Rendering Results

3D 마야 프로그램과 유니티를 활용한 라이팅·렌더링 테스트 결과, 두 프로그램 모두 비슷한 라이팅 타입과 속성을 가지고 있어 근접한 퀄러티의 결과물을 만들어 낼 수 있었다.

마야에서는 실내 배경에 총 22개의 라이트를 세팅했는데, Directional light로 창에서 들어오는 달빛을, 17개의 Area light를 rectangle shape, disk shape 등으로 모양을 바꿔가며 벽난로와 창가, 천장과 실내등을 조명했다. 4개 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 41분 59초가 걸렸다. 라이트를 세팅한 작업시간은 약 1시간 반이 소요되었다.

Table 10. Comparison of Lighting and Rendering Results

Item	Maya	Unity
No. of lights and Types	·1 Directional light ·17 Area lights ·4 Mesh lights	·1 Directional light ·5 Area lights ·4 Mesh lights ·8 Point lights
Total lights	22 ea	18 ea
Shader	Blinn	Standard Shader
Render Sample Setting	·Camera(AA) 6 ·Diffuse 5 ·Specular 3 ·Transmission 2 ·SSS 2 ·Volume Indirect 2	·Direct Sample 128 ·Indirect Sample 256 ·Environment Sample 512 ·Lightmap Resolution 32
Working Time in Lighting	Approximately 1.5 hour	Approximately 1 hour
Bake Time	Not Applicable	7 min 48 sec
Render Time	41 min 59sec	Not Applicable

유니티에서는 마야에서 모델링된 실내 배경을 임포트해 라이팅 작업을 시작했다. Directional light로 창에서 들어오는 달빛을, 5개의 Area light와 6개의 Point light로 실내등을 조명했다. 4개의 mesh light는 천장의 전구로 세팅했다. 총 18개의 라이트가 사용되었다. 쉐이더는 행렬 연산이 자동으로 되는 Surface Shader를 적용하여 노말맵(normal map)과 metalness를 적용해 질감을 살렸다. 라이팅 작업 시간은 약 1시간이 걸렸다. 라이트를 배치할 때마다 테스트 렌더를 할 필요가 없이 바로 확인이 가능해 마야보다 작업 시간이 절감되었다. 단지 라이트매핑을 베이크 하는데 7분 48초 정도의 시간이 추가되었다. 렌더링 시간은 불필요했다.

V. Conclusions

최근의 게임엔진이 풀 퀄러티 프리비주얼라이제이션(Full Quality Previsualization), 리얼타임 퍼포먼스 캡처(Realtime Performance Capture) 등 그래픽 기능이 발전되면서 게임뿐만 아니라 영화, 애니메이션, 건축, 교육 등 활용 영역을 확장하고 있다. 국내외 프로덕션들도 실시간렌더엔진인 게임엔진에 큰 관심을 가지고 프로젝트에 도입하려는 시도가 이어지고 있다.

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 유니티를 활용하여 렌더링을 비교 분석했다. 실험 결과 마야에서 제작한 에셋을 손쉽게 유니티로 불러올 수 있고, 라이팅 세팅 또한 마야와 비슷한 속성을 가지고 있어 기존 3D 파이프라인에 익숙한 사용자도 큰 부담 없이 접근할 수 있었다.

폴리건의 수가 많은 모델링 데이터나 합성이 필요한 하이 퀄러티의 프로젝트는 마야나 기존 3D 프로그램으로 시퀀스(sequence)로 렌더하여 합성하는 것이 퀄러티 측면에서 유리할 것으로 파악된다. 그러나 TV시리즈 같이 복잡한 합성이 필요 없는 프로젝트의 경우는 유니티도 비슷한 퀄러티의 결과물을 만들어내는 것을 확인했다.

분석 결과, 유니티와 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 부각되었다. 기존 파이프라인과의 대치, 새로운 작업환경에 적응, 게임엔진을 다루는 비게임분야 전문가 부재 등 여러 가지 어려움이 있음에도 불구하고 가까운 미래에 비-게임엔진 분야의 주요 렌더러로 도입될 것을 예상하는 이유가 바로 이러한 장점 때문이다. 이번 실험을 통해 유니티의 라이팅 세팅을 어떤 식으로 할 것인지, 마야의 라이팅·렌더링과 비교해 렌더 타임, 노이즈 퀄러티, 베이크, 렌더 방식 등을 분석해 봄으로써, 기존 3D 애니메이션 파이프라인에 익숙한 아티스트들에게 실시간렌더엔진에 접근할 수 있는 가이드를 제

공했다는 데에 의의가 있다. 기존 파이프라인에 익숙하고, 프로덕션의 R&D가 부족한 상황에서 선뜻 실시간렌더엔진의 도입을 망설이고 있는 상황에서 이번 연구가 도움이 되기를 바란다. 향후 연구 계획은 유니티와 더불어 게임엔진 분야에서 양대 산맥을 이루고 있는 언리얼엔진의 렌더링을 측정해 비교 분석할 필요가 있을 것으로 판단된다.

REFERENCES

[1] R. GeethaRamani and I. Arun Nivas, "A rendering pipeline framework for photorealistic rendering of animated virtual objects into real scenes," 2014 International Conference on Recent Trends in Information Technology, pp. 1-6, 2014, DOI: 10.1109/ICRTIT.2014.6996125.

[2] Dmlee. "Real-time visual production using unity 3D," Hansea University, p.2, 2019.

[3] J. Broderick, J. Duggan and S. Redfern, "Using game engines for marine visualisation and collaboration," 2016 International Conference on Image, Vision and Computing (ICIVC), pp.96-101, 2016, DOI: 10.1109/ICIVC.2016.7571280.

[4] From DOOM to unia : History of Game Engine, https://sergeswin.com/374.

[5] Maya site, https://www.autodesk.co.kr/.

[6] SsYang, “A study on the System Process of Production pipeline of 3D animation,” Proceedings of the Korea Contents Association Conference, pp.198-202, Seoul, Korea, May 01 2008.

[7] Chjung,Mjkim, “Comparative Analysis for Production Pipeline Between 3D Animation and Machinima,“ Journal of Korea Entertainment Industry Association, 10(2), pp.313-321. April 2016, DOI : 10.21184.

[8] Ryu Jae Ho, "Proposal of essential components for construction of architectural digital twin using game engine,” ARCHITECTURAL INSTITUTE OF KOREA, 41.1 pp.518-519, April 2021.

[9] 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), pp. 236-239, 2019, DOI: 10.1109/ICICTA49267.2019.00057.

[10] Suh, Dong Hee, “A Study of a 3D Animation Production Pipeline Using a Game Engine” . CONTENTS PLUS, 19(5), pp.71-84, 2021.

[11] MkKim, “the Game Graphics”, Viel Books, p.98, 2015.

[12] T.Whitted, "Origins of Global Illumination," in IEEE Computer Graphics and Applications, vol. 40, no. 1, pp. 20-27, 1 Jan.-Feb. 2020, DOI: 10.1109/MCG.2019.2957688.

Authors

Gum-Young Kim received the MFA degree in the Department of Computer Animation from the Hongik University, and majored VFX and Computer Animation at Sheridan College in Canada.

She worked on Academy Award Feature Animation “Rango”, “Starwars;Clonewars“ Season 1,2,3 and many others. Currently working as a full time lecturer at Seowon University. She is interested in real-time lighting and rendering.

Byong-Kwon Lee received the B.S., M.S. and Ph.D. degrees in Computer Science and Engineering from Hanbat, Hannam and Chungbuk University Korea, in 2000, 2003 and 2007, respectively.

His main areas of interest are embedded systems, virtual and augmented reality, and artificial intelligence. The field currently being studied is the construction of an exhibition hall using virtual reality. It is a technology that combines AI with cultural uniform restoration technology as a future research field.