한국컴퓨터그래픽스학회
Korea Computer Graphics Society Vol. 25, No. 4, P. 17~24

유니티와 언리얼 엔진 4의 구조와 구현 방식 비교 연구

이한성[1] 류승택[1] 서상현[2*]

[1]한신대학교 컴퓨터공학과

[2]중앙대학교 컴퓨터예술학부

jkhanseong@naver.com, stryoo@hs.ac.kr, sanghyun@cau.ac.kr

A Comparative Study on the Structure and Implementation of Unity and Unreal Engine 4

HanSeong Lee[1] SeungTaek Ryoo[1] SangHyun Seo[2*]

[1]Department of Computer Engineering, Hanshin University, Yangsan-dong, Osan, Gyeonggi 447-791, Korea

[2]School of Computer Art, College of Art and Technology, Chung-Ang University, Anseong, Republic of Korea

요 약

게임 산업에서 전통적인 게임 개발은 자체 엔진을 사전에 개발하여 게임 콘텐츠를 완성하는 방식이었다. 이로 인해 게임을 개발할 때마다 자체 게임엔진을 개발하는 비용부담이 크다는 단점을 가지고 있다. 또한, 게임 콘텐츠와 게임엔진이 독립적인 형태로 개발이 되어야 하지만 게임엔진이 사전에 마련되지 않으면 게임 콘텐츠가 개발되지 않는 문제점을 갖는다. 이러한 문제점을 해결하기 위해 최근 게임 산업에서는 게임엔진 구현에 집중하기보다는 게임 콘텐츠 생산성을 높이는 방법인 상용 엔진의 라이선스를 구매하여 개발하는 방식을 선택하고 있다. 본 논문에서는 컴포넌트 기반 시스템(Component-Based System)이고 시장 점유율이 가장 높은 게임엔진인 유니티와 언리얼 엔진을 대상으로 두 엔진의 구조, 에디터 인터페이스, 그리고 지원하는 개발 언어를 비교하여 각 게임엔진이 가진 특징들을 분석한다. 특히, FPS(First-Person Shooter) 게임의 구현을 통해 두 엔진에서의 플레이어 구성과 이벤트 처리 방식의 차이를 비교 설명하였다.

Abstract

In the game industry, traditional game development way was to develop a game engine in advance before developing game content. This has the disadvantage is the high cost of developing the game engine. Further, there is a problem that the game content are not developed unless the game engine is prepared in advance. In order to solve these problems, the game industry has recently selected a method of purchasing a license for a commercial engine, which is a method for improving the game content development productivity, rather than concentrating on game engine development. In this paper, we describe a game object and the structure in Unity and Unreal Engine 4. We compared the editor interface and the supported development languages in the two game engines. We also implemented a First-Person Shooter game to compare how the player's configuration and event handling differ from the two game engines.

키워드: 게임엔진, 게임 개발 도구, 1인칭 슈팅 게임, 유니티, 언리얼 4

Keywords: Game engine, Game development tools, FPS(First-Person Shooter), Unity, Unreal Engine 4

*corresponding author: SangHyun Seo/Chung-Ang University(sanghyun@cau.ac.kr)

Received : 2019.08.13./ Review completed : 1st 2019.08.22. / Accepted : 2019.08.27.
DOI : 10.15701/kcgs.2019.25.4.17
ISSN : 1975-7883(Print)/2383-529X(Online)

1. 서론

게임 산업에서 전통적인 게임 개발은 자체 엔진을 사전에 개발하고, 그것을 기반으로 게임 콘텐츠를 채워가는 방식을 선택했다. 이로 인해 게임을 개발 시 게임 콘텐츠보다 자체 게임엔진 개발을 위한 비용부담이 크다는 단점을 가지고 있다. 또한, 게임 콘텐츠와 게임엔진이 독립적인 형태로 개발이 되어야 하지만 게임엔진이 사전에 마련되지 않으면 게임 콘텐츠가 개발되지 않는 문제점을 갖는다. 이러한 문제점을 해결하기 위해 최근 게임 산업에서는 자체적으로 3D 게임엔진을 개발하기보다는 개발의 편의성을 높이기 위해 상용 게임엔진의 라이선스를 구매하여 개발하는 방식을 선택하고 있다[1].

초기 상용 게임엔진은 둠 엔진으로 1993년 이드 소프트웨어에서 개발하였다. 둠 엔진은 2D 엔진으로 스프라이트, 객체, 캐릭터 외에도 전체 레벨 맵을 구성할 수 있었다[2].

현재 상용 게임엔진은 유니티, 언리얼, 크라이엔진, Cocos2d 등 다양하며 2D와 3D를 모두 지원하는 형태로 발전했다. 본 논문에서는 주된 게임엔진인 유니티(Unity)와 언리얼 엔진 4(UE4; Unreal Engine 4)를 대상으로 두 엔진의 구조에 관해서 설명한다[3]. 그리고 게임 장르 중 하나인 FPS(First-Person Shooter) 게임을 구현하여 유니티와 언리얼 엔진에서의 플레이어 구성과 이벤트 처리 방식에 대해 비교함으로써 각 엔진의 특징들을 파악하고자 한다.

2. 관련 연구

유니티(Unity)[4]는 3D 및 2D 비디오 게임의 개발 환경을 제공하는 게임엔진이면서 3D 애니메이션, 건축 시각화, 가상현실 등 반응형 콘텐츠 제작을 위한 통합 저작 도구이다. 반면 언리얼 엔진(Unreal Engine)[5]은 미국의 에픽 게임즈에서 개발한 3차원 게임엔진이다. 1994년부터 현재까지 꾸준한 개량을 통해 발전되고 있으며, 수십 개의 비디오 게임에 사용되고 있는 미들웨어 솔루션이다. 언리얼 엔진의 버전은 세대별로 구분을 하고 있으며 현재까지 4세대까지 출시되었다.

현재 게임엔진과 관련하여 다양한 연구가 이뤄지고 있다. 이러한 연구에서는 게임엔진에 대한 의미와 구성 요소 그리고 기능에 관한 내용은 물론 게임엔진의 간단한 역사와 현대 게임엔진이 되기까지의 발전 과정하고 있으며[2], 상용 게임엔진을 게임 개발이 목적이 아닌 군사 훈련에 적합한 게임엔진을 비교하여 기술하고 있다[6].

기존 게임엔진에 대한 비교는 주로 유니티와 언리얼 엔진의 기능에 대해 다루고 있으며, 그중 팩맨 게임을 구현하여 유니티와 언리얼 엔진에서의 렌더링 결과 및 성능을 비교한 연구가 있다[7]. 해당 연구에서는 경로 찾기, 물리 엔진, 애니메이션, 스크립팅 그리고 다양한 렌더링 스타일에 대해서 벤치마킹하여 두 가지 엔진을 비교 설명하고 있다.

다른 연구결과로는 데스크톱과 모바일 기기용 게임엔진을 비교 분석한 연구가 있다[3]. 해당 연구에서 유니티와 언리얼 엔진 4를 이용해 공식 튜토리얼을 기반으로 안드로이드용 슈팅 게임을 개발하여 두 가지 엔진을 비교 설명하였다.

이미 많이 활용되고 있는 두 게임엔진이고, 두 게임엔진 모두 컴포넌트 기반 시스템을 사용하고 있으나, 그 접근법에 있어 개발 언어, 인터페이스, 라이선스 정책 등 구조적인 차이를 보임에도 두 엔진에 대한 비교는 경험자들에 의한 구술 정도일 뿐, 구체적으로 정리된 경우는 부족한 것이 현실이다.

본 연구에서는 다양한 측면에서 두 게임엔진을 비교 설명하고, 간단한 FPS 게임제작을 통해 두 게임엔진의 특징들을 구체적으로 비교함으로써, 게임을 개발하고자 하는 사용자들에게 다양한 사전 정보를 제공하고자 한다.

3. 인터페이스 및 개발 언어 비교

3.1 에디터 인터페이스

유니티의 에디터 인터페이스는 툴바(Toolbar), 계층(Hierarchy), 씬(Scene), 인스펙터(Inspector), 프로젝트(Project)로 구성되며 Figure 1에서 보여주고 있다.

언리얼 엔진의 에디터 인터페이스는 툴바(Toolbar), 모드(Modes), 뷰포트(Viewport), 월드 아웃라이너(World Outliner), 디테일(Details), 콘텐츠 브라우저(Content Browser)로 구성되며 Figure 2와 같다.

Figure 1. Editor Interface in Unity

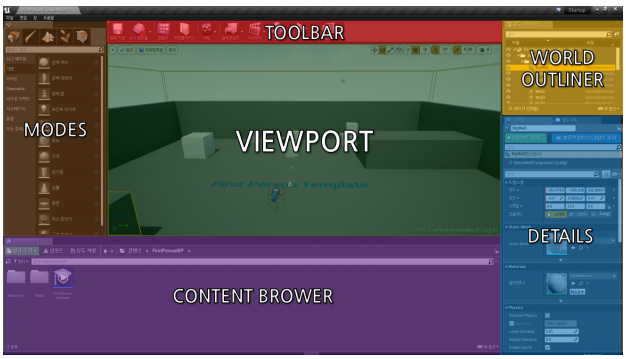

Figure 2. Editor Interface in Unreal Engine 4

에디터의 레이아웃을 설정할 수 있는 기능을 두 에디터에서 모두 제공한다. 인터페이스의 기능은 유사하지만 사용되는 용어가 다르다. 두 에디터 인터페이스 용어를 비교한 것이 Table 1이다. 용어 외에도 배치하는 방법에도 차이가 있다. 유니티의 오브젝트 배치하는 방법은 계층 창에서 드롭다운 버튼을 통해 추가하는 방식이지만 언리얼 엔진은 모드 창에 있는 오브젝트를 선택하여 뷰포트로 드래그 앤 드롭하는 방식으로 한다는 차이점이 존재한다.

Table 1. The Comparison of Unity and UE4 Editor Interface

Name	Unity	Unreal Engine 4
Game World	Scene	Viewport
Object Hierarchy	Hierarchy	World Outliner
Asset Library	Project	Content Browser
Object Properties	Inspector	Details

3.2 개발 언어

유니티에서는 개발 언어로 자바스크립트와 유사한 유니티 스크립트를 제공했다. 하지만 3.6%의 낮은 사용률과 새로운 스크립트와 관련된 기능을 제공하는데 저해된다는 이유로 유니티 2017.2 베타 버전부터 자바스크립트 생성 항목을 제거하였다. 그래서 현재 유니티는 개발 언어로 C#만 지원하고 있다[8].

유니티에서 제공하는 C#은 고수준 언어에 해당하며 모든 변수의 메모리 관리는 가비지 컬렉터(Garbage Collector)에 의해 관리된다. 포인터도 사용하지 않아 언리얼 엔진에서 제공하는 C++와 비교하여 고려사항이 적다.

언리얼 엔진의 경우 개발 언어로 C++와 블루프린트를 사용할 수 있다. 그리고 언리얼 엔진에도 메모리 관리가 존재한다. 언리얼 엔진의 메모리 관리 방식은 게임 오브젝트의 부모 클래스인 UObject 클래스에 참조 횟수 계산 방식의 메모리 제어 기법을 적용하였다. 그러므로 언리언 엔진에서 생성되는 오브젝트는 프로그래머가 직접 제거하지 않아도 참조가 없을 때 자동으로 소멸한다. UObject 클래스를 상속받지 않는 오브젝트에 대해서는 프로그래머가 직접 생성과 소멸을 관리할 수 있고 스마트 포인터를 적용할 수도 있다.

언리얼 엔진에는 C++뿐만 아니라 블루프린트를 지원하며 블루프린트는 언리얼 에디터 안에서 노드 기반의 인터페이스를 사용하여 게임플레이 요소를 만드는 개념을 토대로 한 비주얼 스크립팅 시스템(Visual Scripting System)이다[9].

블루프린트는 C#과 C++와 달리 텍스트 기반이 아니기에 문법에 대한 지식이 필요하지 않다. 그래서 프로그래머보다 디자이너나 기획자가 주로 사용한다. 하지만 기능이 많거나 로직이 복잡해지면 텍스트 기반의 프로그래밍보다 효율성이 떨어진다는 단점을 가지고 있다. 그러므로 대부분 복잡하고 큰 기능을 프로그래머가 C++로 구현하고 이를 상속받아 디자이너나 기획자가 블루프린트로 구성한다.

두 엔진에서 사용되는 개발 언어를 프로그래밍 관점에서 비교한 결과가 Table 2이다.

Table 2. The comparison of Unity and UE4 script languages

	C#	C++	Blueprint
Game Engine	Unity	UE4	
Programming	Text-Based		Visual-Based
User	Programmer		Designer, Planner
Memory Management	Automatic	Manual / Automatic	Automatic
Pointer	Not Used	Used	Not Used
Difficulty	Normal	Hard	Easy

3.3 라이선스

유니티의 라이선스는 구독 방식으로 Personal, Plus, Pro 3가지가 있다. 연간 수익 또는 자본금이 $100,000 이하일 때 Personal 옵션을 사용하여 무료로 사용할 수 있다. Plus와 Pro는 매월 구독료를 내며 Personal과 달리 별도로 제공되는 혜택과 엔진에서 추가로 제공되는 기능이 존재한다. 언리얼 엔진의 라이선스는 무료이다. 하지만 상용 제품에서 분기당 게임별 $3,000를 초과한 경우 총 수익에서 5%의 사용료를 에픽 게임즈에 지불해야 한다[10].

4. 게임엔진의 구조와 구현 방식 비교

4.1 오브젝트 개념

유니티의 씬은 여러 개의 게임 오브젝트(Game Object)로 구성되며 게임 오브젝트를 통해 상호작용이 일어난다. 유니티의 게임 오브젝트 구조를 도식화한 것이 Figure 3이다. 구조를 보면 게임 오브젝트는 하나의 트랜스폼(Transform)을 가지며, 추가로 여러 개의 컴포넌트(Component)를 포함할 수 있다. 트랜스폼은 오브젝트의 위치, 회전, 크기, 부모-자식 상태를 저장한다.

언리얼의 레벨은 여러 개의 액터(Actor)로 구성되며 액터를 통해 상호작용이 일어난다. 언리얼의 게임 오브젝트인 액터를 도식화한 것이 Figure 4이다. 액터는 하나의 루트 컴포넌트(Root Component)라는 이름으로 씬 컴포넌트(Scene Component)를 가진다. 추가로 여러 개의 액터 컴포넌트(Actor Component)를 포함할 수 있다. 씬 컴포넌트는 유니티의 트랜스폼과 유사하며 오브젝트의 위치, 회전, 크기, 부모-자식 상태를 저장한다.

언리얼에는 클래스 명에 사용하는 2가지 접두사가 있다. 접두사 'A'는 스폰이 가능한 게임플레이 오브젝트의 베이스

클래스에서 확장되며, 액터로 월드에 스폰할 수 있다. 접두사 'U'는 모든 게임플레이 오브젝트의 베이스 클래스에서 확장된다. 접두사 'U'가 붙은 클래스는 월드에 스폰할 수 없고 액터에 속해야 한다. 일반적으로 컴포넌트와 같은 오브젝트가 해당한다[11].

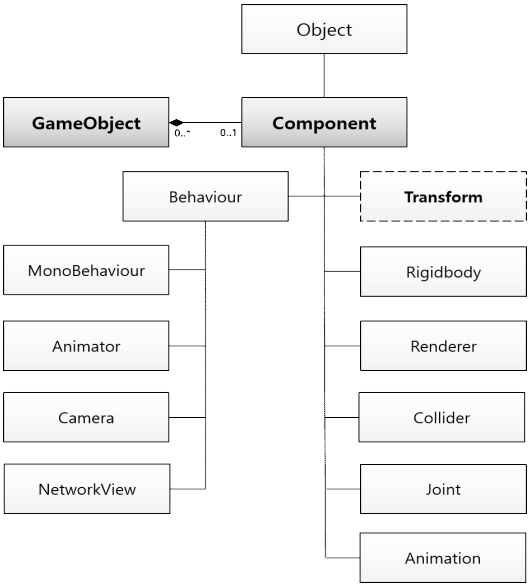

Figure 3. Component Structure in Unity

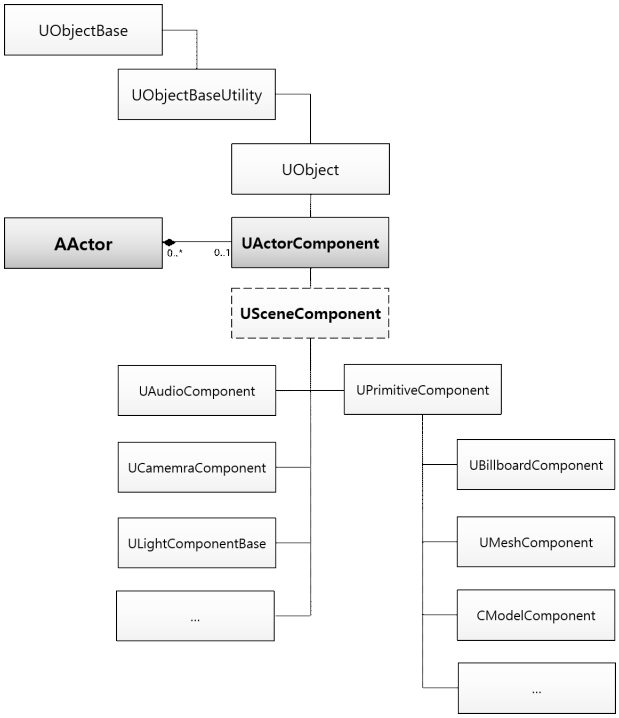

Figure 4. Component Structure in UE4

유니티와 언리얼 엔진의 게임 오브젝트는 하나의 게임 오브젝트에 여러 개의 컴포넌트를 포함하는 컴포넌트 기반 시스템(Component-Based System)이다. 이러한 시스템의 장점은 게임 오브젝트와 행동 간의 결합도가 낮아지면서 게임 오브젝트 시스템이 높은 유지 보수성과 유연성을 갖는다[12].

4.2 게임 오브젝트 상호작용

유니티와 언리얼 엔진에서 같은 FPS 게임을 구현하면서 플레이어의 구현과 이벤트 처리가 두 게임엔진에서 어떤 차이점을 갖는지 비교한다. 게임을 구현하기 위해 Tommy Tran의 게임 콘텐츠[13]를 사용하였다. 같은 게임 콘텐츠를 사용하여 유니티와 언리얼 엔진에서 FPS를 구현한 결과가 Figure 5와 6이다.

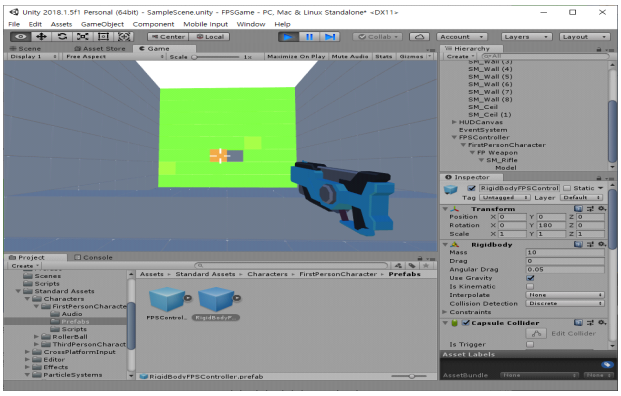

Figure 5. A Simple FPS in Unity

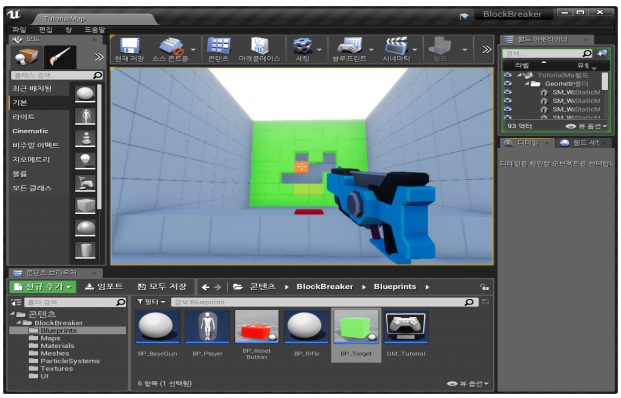

Figure 6. A Simple FPS in UE4

4.2.1 플레이어 구성

유니티는 플레이어를 프리팹(Prefab)으로 제공하고 있다. 그러므로 에디터 내에 오브젝트로 생성하지 않고 임포트 패키지(Import Package)를 사용하여 기본 애셋(Standard Assets)을 불러와 구성한다. "FPSController.prefab"과 "RigidBodyFPSController.prefab" 두 개의 1인칭 캐릭터 컨트롤러 프리팹을 제공하고 First Person Controller 컴포넌트가 공통으로 있다. "FPSController.prefab"은 캐릭터 컨트롤러로 구현한 방식이다.

Figure 7은 캐릭터 컨트롤러의 속성이며 캐릭터 이동과 관련하여 다양한 기능이 포함되어 있다. 경사 제한을 설정할 수 있어 경사를 만났을 때 이동할 수 있는 여부와 플레

이어 반경이나 높이에 대한 부가적인 정보를 받아 빠르게 플레이어를 구성할 수 있다.

"RigidBodyFPSController.prefab"은 강체(RigidBody)로 구현한 방식으로 캐릭터 컨트롤러를 사용하는 방법에 비교해 제공하는 기능이 상대적으로 적다. 그래서 경사에 대한 처리와 같은 부가적인 기능이 없다.

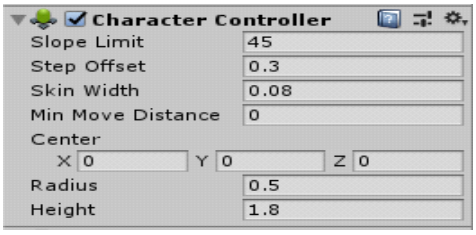

Figure 7. Character Controller in Unity

언리얼 엔진은 게임 오브젝트로 플레이어를 제공하고 있어 애셋(Assets)을 불러올 필요가 없다. Figure 8에 보이는 공백 캐릭터를 가지고 플레이어를 생성할 수 있다. 공백 캐릭터에는 캐릭터 이동(Character Movement) 컴포넌트가 Figure 9와 같이 포함되어 있다. 해당 컴포넌트를 통해 플레이어 움직임이 처리된다.

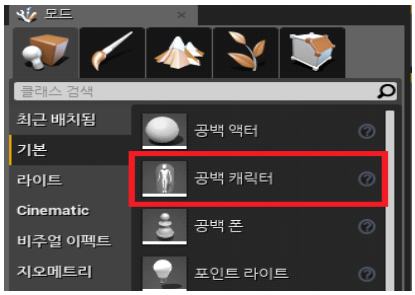

Figure 8. Empty Character in UE4

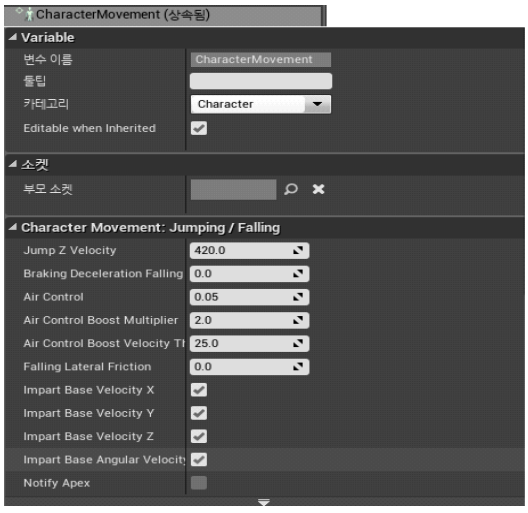

Figure 9. Character Movement in UE4

유니티와 언리얼 엔진의 캐릭터 이동 기능을 비교하면 Table 3과 같이 정리할 수 있다. 유니티의 경우 이동은 걷기, 뛰기, 점프, 낙하 시 처리가 가능하다. 부가적으로 사운드와 카메라 애니메이션 처리를 제공한다. 언리얼 엔진의 경우 뛰기를 제외한 캐릭터의 모든 이동을 설정할 수 있다. 네트워크와 인공지능과 같은 고급기능도 제공한다. 하지만 유니티에서 제공하고 있는 사운드와 카메라 애니메이션 기능은 포함하고 있지 않아 추가 구현해야 한다.

Table 3. The Comparison of Unity and UE4 Character Component

Function	First Person Controller (Unity)	Character Movement (UE4)
Walking	O	O
Run	O	X
Jumping/Falling	O	O
Swimming	X	O
Flying	X	O
Rotation Settings	X	O
Networking	X	O
Physics Interaction	X	O
Avoidance	X	O
Nav Mesh Movement	X	O
Sound	O	X
Mouse Look	O	X
Fov Kick	O	X
Head Bob	O	X
Jump Bob	O	X

유니티의 경우 캐릭터 프리팹을 사용하면 바로 게임을 플레이할 수 있다. 하지만 언리얼 엔진의 경우 공백 캐릭터를 배치해도 캐릭터가 움직이지 않는다. 캐릭터를 움직이기 위해서는 설정에서 입력키 맵핑을 한 다음 입력 이벤트를 Figure 10과 같이 캐릭터 이동 컴포넌트와 연결해야 플레이어를 조작할 수 있다.

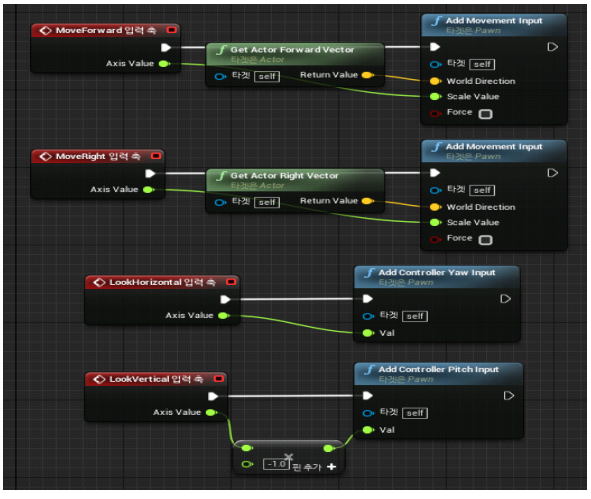

Figure 10. Key Events Connection with Character Movement Component in UE4 (Blueprint)

4.2.2 이벤트 처리

유니티와 언리얼 엔진의 초기화, 해제, 업데이트, 트리거 이벤트 함수 처리 방식은 같다. 두 게임엔진에서 사용되는 초기화, 종료, 업데이트, 트리거 이벤트 함수의 이름을 Table 4와 같이 정리할 수 있다. 이벤트 구현 방식의 차이점이 있는데, 유니티의 경우 게임 오브젝트 클래스가 아닌 MonoBehaviour 클래스를 상속받아 이벤트를 작성해야 한다. 하지만 언리얼 엔진에서는 게임 오브젝트인 액터 컴포넌트뿐만 아니라 액터 클래스를 상속받아 이벤트를 작성할 수 있는 특징이 있다.

Table 4. The Comparison of Unity and UE4 Events

Event	Unity	UE4
Initialize	Start	BeginPlay
Finalize	OnDestroy	EndPlay
Update	Update	Tick
Trigger	OnTriggerEnter	ActorBeginOverlap
	OnTriggerExit	ActorEndOverlap

유니티는 입력 이벤트를 별도의 함수로 제공하고 있지 않고 Figure 11과 같이 업데이트 이벤트에서 구현한다. 이와 달리 언리얼 엔진에서는 프로젝트 설정에서 입력을 지정하면 Figure 12와 같이 입력 이벤트를 추가할 수 있다.

언리얼 엔진에서는 유니티와 비교해 다양한 이벤트와 서버에서만 발동하는 서버 측 이벤트를 지원하고 있다는 차이점이 있다. 서버 측 이벤트는 Figure 13에서 보이는 것과 같이 노드의 우측 상단에 아이콘이 표시되어 있어 쉽게 구분할 수 있다. 언리얼 엔진에서 제공하는 서버 측 이벤트를 활용하면 별도의 서버 처리를 구현하지 않아도 멀티플레이어를 구성할 수 있다.

```
// Update is called once per frame
void Update () {
    Shoot();
}

void Shoot() {
    if (canShoot && Input.GetButton("Fire1")) {
        canShoot = false;
        playerGun.GetComponent<BaseGun>().Shoot();
        StartCoroutine(FireRate());
    }
}
```

Figure 11. Key Event in Unity (C# Script)

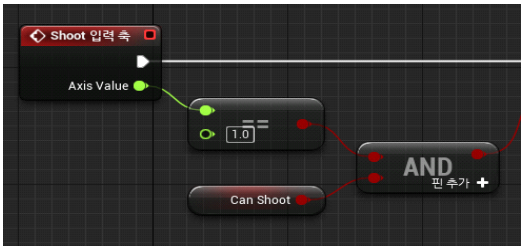

Figure 12. Key Event in UE4 (Blueprint)

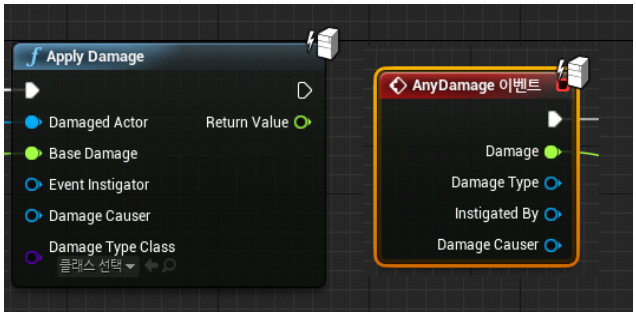

Figure 13. Server Side Events in UE4 (Blueprint)

4.3 사용자 환경 UI

Figure 14는 유니티에서 사용하는 사용자 인터페이스인 UGUI(Unity Graphic User Interface)다. 사용자 인터페이스는 캔버스(Canvas)를 통해 표현된다. 모든 UI 요소는 캔버스의 자식으로 추가하여 구성한다.

Figure 15는 언리얼 엔진에서 사용하는 사용자 인터페이스인 UMG(Unreal Motion Graphics UI Designer)이다. UMG의 경우 사용자 인터페이스를 하나의 위젯 형태로 작업하며 위젯을 실행하면 언리얼 에디터에서 새로운 디자이너 탭이 생성된다. 디자이너 탭을 통해서 위젯을 수정할 수 있다. 그리고 위젯에는 기본적으로 캔버스 패널이 포함되어 있다.

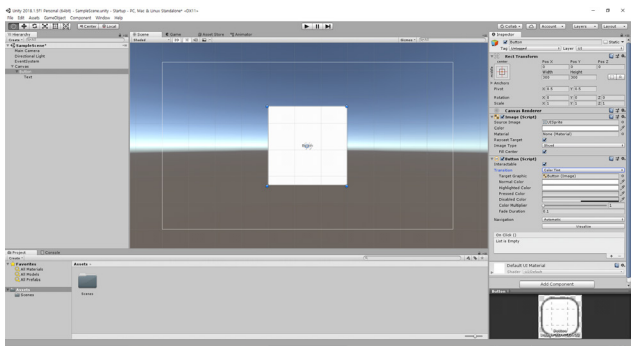

Figure 14. UGUI in Unity

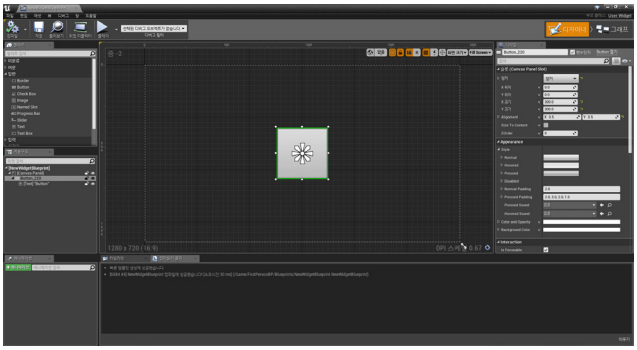

Figure 15. UMG in UE4

UGUI는 캔버스를 생성하면 해당 씬에 바로 사용자 인터페이스가 적용되어 플레이 시 확인할 수 있다. 하지만

UMG의 경우 별도의 위젯이 파일로 생성되며 실제 게임에서 적용된 것을 확인하기 위해서 게임 내 위젯을 생성하는 부가적인 코드 작업이 필요하다.

사용자 인터페이스에 데이터 바인딩하기 위해서는 UGUI는 스크립트를 작성해야 한다. 이와 달리 UMG는 위젯이 블루프린트 기반이다. 그러므로 UMG 창 안에서 필요한 변수를 생성하고 블루프린트를 이용해 바인딩할 수 있다.

5. 결론

본 논문에서는 유니티와 언리얼 엔진의 사용자 인터페이스, 개발 언어, 라이선스 정책 및 게임 오브젝트 개념 및 구조에 관해 비교설명 하였다. 그리고 FPS 게임을 구현하여 플레이어의 구성과 이벤트 처리가 두 게임엔진에서 어떤 차이점을 갖는지 비교하였다.

두 게임엔진은 같은 컴포넌트 기반 시스템이라는 구조를 사용하고 있지만, 게임 오브젝트의 하위 구조에 있어, 유니티 엔진은 사용자 접근이 쉽도록 단순하지만, 언리얼 엔진은 많은 세부 컴포넌트로 나누어져 있음을 볼 수 있었다. 또한, 플레이어 구성의 경우 유니티는 캐릭터를 게임 오브젝트로 제공하지 않고 프리팹으로 제공한다. 언리얼 엔진의 경우 게임 오브젝트로 공백 캐릭터를 지원한다.

각 캐릭터의 컴포넌트 기능에서 캐릭터 이동 기능은 유니티와 달리 언리얼 엔진의 캐릭터 이동 컴포넌트가 네트워크나 인공지능과 같은 고급기능을 지원한다.

이벤트 처리는 두 게임엔진이 모두 유사하다. 하지만 언리얼 엔진에서는 입력 이벤트 함수를 제공하며 서버 측 이벤트를 지원하여 멀티플레이어를 생성할 수 있다. 반면에 유니티의 경우 업데이트 이벤트에 조건을 주는 방식이며 서버 측 이벤트를 별도로 제공하고 있지 않다.

각 게임엔진에는 장단점이 존재하기 때문에 단적으로 특정 게임엔진이 좋다고 말할 수 없다. 그뿐만 아니라 게임엔진을 선택하는 요소에 있어 전문 지식, 팀워크, 리소스 및 시간과 같이 다양한 요소가 있으며 상당히 복잡하기 때문이다[14]. 그렇지만 본 연구에서는 두 게임엔진의 구조와 구현 방식을 소개 및 비교함으로써 사용자에게 어떤 게임엔진이 더 적합한지에 대한 정보를 줄 수 있을 것이다. 그리고 본 연구가 앞으로 게임엔진과 게임 개발 분야 등에 활용될 수 있을 것으로 기대된다.

향후 연구에서는 NPC, 애니메이션, 인공지능에 대해 두 게임엔진의 구조와 구현 방식의 차이점에 대해 연구가 필요하다.

감사의 글

이 논문은 2016년도 정부(교육과학기술부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업임(NRF-2016R1D1A1B03935378)

References

[1] IT DAILY, “The SW industry is looking beyond creative convergence.” http://www.itdaily.kr/news/articleView.html?idxno=92038. 2018.12.01.

[2] P. Paul, S. Goon, and A. Bhattacharya, "History and Comparative Study of Moden Game Engines", *International Journal of Advanced Computer and Mathematical Sciences 3*, pp. 245-249, 2012.

[3] Christopoulou, E., & Xinogalos, S. (2017). “Overview and Comparative Analysis of Game Engines for Desktop and Mobile Devices”, *International Journal of Serious Games*, 4(4), 2017. https://doi.org/10.17083/ijsg.v4i4.194.

[4] Wikipedia, “Unity (Game Engine).” https://en.wikipedia.org/wiki/Unity_(game_engine).

[5] Wikipedia, “Unreal Engine.” https://en.wikipedia.org/wiki/Unreal_Engine.

[6] Mr. Stuart Armstrong, “Game Engine Review”, *NORTH ATLANTIC TREATY ORGANIZATION*, 2013.

[7] Antonín Šmíd, “Comparison of Unity and Unreal Engine”, Czech Technical University in Prague, 2017.

[8] Unity blog, “UnityScript’s long ride off into the sunset.”, 2017.08.11 https://blogs.unity3d.com/kr/2017/08/11/unityscripts-long-ride-off-into-the-sunset/.

[9] Unreal Engine, “Blueprint Introduction.”, https://api.unrealengine.com/KOR/Engine/Blueprints/GettingStarted/index.html.

[10] Alie Hajalie, “Unity vs. Unreal Engine 4”, http://web.eecs.umich.edu/~sugih/courses/eecs441/common/Unity-Unreal.pdf, April, 16, 2015.

[11] Unreal Engine 4 Document, “Gameplay Classes”, https://docs.unrealengine.com/en-US/Programming/UnrealArchitecture/Reference/Classes/index.html

[12] Nicolas Porter, “Component-based game object system”, Carleton University, 2012.

[13] Raywenderlich, “How to Create a Simple FPS in Unreal Engine 4” https://www.raywenderlich.com/228-how-to-create-a-simple-fps-in-unreal-engine-4, 2018.01.18

[14] Petridis, P., Dunwell, I., Panzoli, D., Arnab, S., Protopsaltis, A., Hendrix, M. and Freitas, S., “Game Engines Selection Framework for High-Fidelity Serious Applications”, *International Journal of Interactive Worlds*, Vol 2012, 19 pages, 2012.

〈저자소개〉

이 한 성

- 2017년 : 한신대학교 컴퓨터공학부 (공학사)
- 2018년~현재 : 한신대학교 컴퓨터공학과 석사과정
- 관심분야 : 게임엔진, 게임 개발, 온라인 게임
- https://orcid.org/0000-0002-0175-8118

류 승 택

- 1996년: 중앙대학교 전자계산학과 공학사
- 1998년: 중앙대학교 컴퓨터공학과 공학석사
- 2002년: 중앙대학교 영상공학과 컴퓨터그래픽스전공 공학박사
- 2004년-현재: 한신대학교 컴퓨터공학부 교수
- 관심분야: 비사실적 렌더링, 실시간 렌더링, 감성기반 렌더링
- https://orcid.org/0000-0002-4180-7922

서 상 현

- 1998년: 중앙대학교 컴퓨터공학과 (학사)
- 2000년: 중앙대학교 영상공학과 공학석사
- 2010년: 중앙대학교 영상공학과 공학박사
- 2011년-2013년: 프랑스 리옹 1대학, LIRIS 연구소, 박사후연구원
- 2013년-2016년: 한국전자통신연구원, 선임연구원
- 2016년-2019년: 성결대학교 미디어소프트웨어학부 조교수
- 2019년-현재: 중앙대학교 컴퓨터예술학부, 부교수
- 관심분야: 컴퓨터그래픽스, 가상현실/증강현실, 게임기술, 영상처리 및 컴퓨터비전
- https://orcid.org/0000-0002-4824-3517