JDCS
디지털콘텐츠학회논문지
Journal of Digital Contents Society
Vol. 25, No. 3, pp. 771-779, Mar. 2024

Check for updates

게임 제작을 위한 JSON 라이브러리 성능 비교: 유니티 엔진 중심으로

김 태 환[1] · 경 병 표[2*]
[1]공주대학교 게임디자인학과 박사과정
[2*]공주대학교 게임디자인학과 교수

Performance Comparison of JSON Libraries for Game Development using Unity Engine

Tae-Hwan Kim[1] · Byung-Pyo Kyung[2*]

[1]Ph.D. Student, Department of Game Design, Kongju University, Kongju 32588, Korea
[2*]Professor, Department of Game Design, Kongju University, Kongju 32588, Korea

[요 약]

본 연구에서는 Unity Engine에서 게임 데이터 처리와 관리를 위해 자주 사용되는 JSON 라이브러리인 JsonUtility, LitJson, 그리고 Newtonsoft.Json의 성능을 비교하였다. 가장 중요한 지표인 직렬화 및 역 직렬화 속도에 초점을 맞춰 애플리케이션을 구성하여 측정하였으며, PC와 Smart Phone 플랫폼에서 각 라이브러리의 성능을 평가하였다. 실험 결과, 기기 성능에 따라 각 라이브러리의 성능은 비례했으며, JsonUtility가 PC에서는 4배, 스마트폰에서는 5~6배 빠른 성능을 보였다. 하지만, JsonUtility는 기본적인 데이터 타입만 지원하고 있어 추가 작업이 필요하며, 간단한 기능만 제공되기 때문에 작업 상황에 맞춰 JSON 라이브러리를 선택해야 한다. 본 연구에서 분석한 결과를 참고하여 게임 제작 시 효율적인 데이터 처리와 관리에 대한 자료로 활용되기를 기대한다.

[Abstract]

In this study, we compare the performances of three JSON libraries commonly used for game data processing and management in the Unity Engine: JsonUtility, LitJson, and Newtonsoft.Json. We focus on the key metrics of serialization and deserialization speeds, constructing applications to measure them and evaluating the performance of each library on PC and smartphone platforms. The experimental results show that the performance of each library varies proportionally with device performance, with JsonUtility exhibiting 4 times faster performance on PCs and 5-6 times faster performance on smartphones. However, JsonUtility supports only basic data types and requires additional work, offering only simple functionality, necessitating the selection of JSON libraries based on specific task requirements. We hope that the findings from this study can serve as a valuable resource for efficient data processing and management in game development.

색인어 : 게임, 데이터 표현, 라이브러리, 성능 비교, 유니티 엔진,

Keyword : Game, JSON, Library, Performance Comparison, Unity Engine,

http://dx.doi.org/10.9728/dcs.2024.25.3.771

Received 05 February 2024; **Revised** 23 February 2024
Accepted 13 March 2024

***Corresponding Author; Byung-Pyo Kyung**

Tel: +82-41-850-0352
E-mail: kyungbp@kongju.ac.kr

Ⅰ. 서 론

게임 제작에 있어 데이터 처리와 관리는 매우 중요한 요소 중 하나이다. 게임은 이미지, 음향, 영상, 3D 모델 등의 다양한 종류의 리소스를 사용하기 때문에, 이러한 데이터들을 체계적으로 관리하고 효율적으로 사용할 수 있어야 한다. 게임은 제작과정에서 수치 데이터의 수정과 조정이 지속해서 이루어지기 때문에, 데이터를 효율적으로 관리하면 로딩 시간을 줄일 수 있고, 게임 내의 밸런스 조정이나 콘텐츠 추가 작업 등의 최적화가 쉽고, 효율적인 다국어 지원 작업이 가능하다. 이처럼 데이터 처리와 관리는 중요하며, 최적화를 통하여 사용자들에게는 부드러운 게임 경험을 제공한다.

게임 산업은 기술의 발전과 함께 지속해서 성장하고 있고, 사용자들은 더욱 현실적이고 흥미로운 경험을 원하기 때문에, 게임 제작은 점점 더 방대하고 복잡한 데이터를 다룰 수밖에 없다. 이러한 이유로 데이터 처리와 관리는 점점 더 중요한 요소로 자리 잡고 있는데, 데이터 교환 형식 중 하나인 JSON (JavaScript Object Notation)이 현재 게임 제작에서 많이 사용되고 있다. JSON은 경량화된 데이터 형식으로서, 구조화된 데이터를 간결하고 효율적으로 표현하는데 적합한 포맷이므로, 게임에서는 JSON을 사용하여 캐릭터 속성, 게임 설정, 임무 수행 등 다양한 정보를 저장하고 전달하는 데 활용된다.

과거에는 상용 게임엔진의 고비용으로 인하여 일반적으로 게임 제작사가 게임엔진을 직접 설계 및 제작하여 게임을 만들어서 출시했던 반면, 현재는 대부분 무료로도 이용할 수 있고, 비교적 저렴한 가격으로 상용 게임엔진을 활용하여 제작이 가능해졌다. 국내에서 많이 활용되고 있는 대표적인 상용 게임엔진으로는 Unity Technologies의 Unity Engine과 Epic Games의 Unreal Engine을 들 수 있다. Unity Engine은 비교적 다루기 쉽고, 가벼우므로 간단한 조작으로 짧은 시간 할 수 있는 캐주얼 게임(Casual Game) 제작에 많이 사용되고 있으며, 점유율이 48%에 달하고, Unreal Engine은 그래픽 표현과 성능에 최적화되어 고품질 게임 제작에 많이 활용되어 최근 13%에서 점유율이 조금씩 올라가고 있다[1]. 그림 1은 2022년 5월 기준 게임엔진별 스팀에 등록된 게임 수를 나타내고, Unity Engine을 활용하여 제작하는 게임이 많음을 알 수 있다.

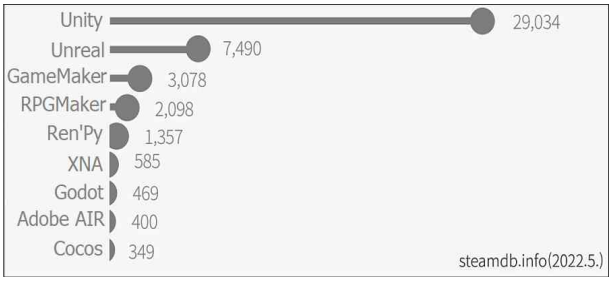

그림 1. 게임엔진별 스팀 등록 게임 (2022년 5월 기준)[2]
Fig. 1. Number of games registered on steam by game engine (as of May 2022)

이처럼 게임 제작에 많이 활용되고 있는 Unity Engine에서도 효율적인 데이터 처리와 관리를 위한 방법으로 주로 JSON 라이브러리를 활용하는데, 이러한 다양한 종류의 JSON 라이브러리는 객체에 저장된 데이터를 출력을 위해 연속적인 데이터로 변환하는 직렬화와 직렬화된 데이터를 JSON 형식으로 변환하는 역 직렬화의 속도가 가장 중요한 요소 중 하나이다. 또한, 각자의 특징과 성능을 가지고 있어, 게임 제작 시에는 상황에 맞게 최적의 라이브러리를 선택해야 한다. 그런데도 현재 관련 자료나 연구는 찾기 어렵고, 게임 제작 시에 대부분 JSON 라이브러리의 성능이 유사할 것으로 판단하여 익숙하거나, 사용하기 쉬운 라이브러리를 선택하여 제작하는 경우가 많다. 이에 본 연구에서는 많은 종류의 JSON 라이브러리 중에 자주 사용되는 JsonUtility, LitJson 및 Newtonsoft.Json들의 직렬화, 역 직렬화의 속도를 게임 서비스 플랫폼으로 많이 선택되고 있는 PC 기반 Windows 환경과 모바일 Smart Phone 기반의 Android 환경으로 나눠 성능을 측정 비교 분석하여 게임 제작 시에 JSON을 활용한 데이터 처리와 관리에서 좀 더 효율적인 방향을 제시하는 것을 목적으로 하였다.

Ⅱ. 관련 연구

2-1 선행 연구

게임 제작에서 일반적으로 데이터 처리와 관리를 위한 방법으로 파일 시스템과 데이터베이스를 많이 사용하는데, 데이터베이스의 경우는 중요한 사용자 정보나 아이템 등을 관리하는 데 사용되고, 간단한 게임 내의 객체에 대한 정보는 텍스트, 바이너리, XML, JSON과 같은 파일 시스템이 주로 사용된다. 관련 연구로는 김현진[3]의 효율적인 게임 파일 I/O를 위한 데이터 형식에 관한 연구가 있는데, Binary, JSON, XML을 비교한 결과 파일 용량이 중요시되는 게임이나 데이터를 빈번하게 전송해야 하는 게임에서는 Binary 형식이 효율적이고, Save와 Load를 많이 해야 하는 게임에서는 JSON 형식이 효율적이라는 결론을 얻었다. 최훈민[4]은 데이터 전송속도 비교연구에서 데이터베이스와 XML, JSON으로 재구성된 파일의 로딩 속도를 비교한 결과 XML과 JSON이 22배 빠르다는 결과를 확인했고, 오진수, 송창기[5]는 XML과 JSON의 데이터 전송속도 비교연구에서 JSON이 37% 데이터 감소 효과와 전송속도 또한 좀 더 나은 성능향상을 보인다는 결과를 얻었다.

2-2 JSON

JSON은 가벼운 DATA 교환 형식이다. 텍스트를 사용하기 때문에 사람이 쉽게 이해하고, 읽고 쓸 수 있는 장점이 있다.

1999년 12월 Standard ECMA-262 3rd Edition의 일부에 기반으로 JavaScript 프로그래밍 언어에서 시작되었으며, 2013년 ECMA-404 The JSON Data Interchange Standard로 표준화되었다. C, C++, C#, Java, JavaScript, Perl, Python 등의 C 계열 언어에 익숙한 텍스트 형식이기 때문에, JSON은 이상적인 데이터 교환 언어로 취급된다. 특정한 플랫폼과 프로그래밍 언어에 독립적이기 때문에 다양한 프로그래밍 언어에서 사용할 수 있으며, 서로 다른 시스템 간에 데이터를 교환하기에 좋고, JavaScript의 문법을 채용했기 때문에, JavaScript를 사용하는 웹 환경에 자주 사용된다. JSON은 두 가지의 기본 구조로 되어 있다. 첫 번째는 name/value 구조의 컬렉션 형태인 객체이고, 다양한 프로그래밍 언어에서 object, struct, record, hash table, dictionary 등 키가 있는 리스트나 연관 배열로 구현된다. 두 번째는 값들의 순서화된 리스트이다. array, list, vector, sequence 등으로 구현된다. 이러한 것들은 보편적인 데이터 구조를 가지며, 대부분의 프로그래밍 언어들은 어떤 형태로든 이를 지원한다. object는 name/value 쌍들의 비 순서화된 집합이다. object는 좌 중괄호({)로 시작하여 우 중괄호(})로 끝내며, name 뒤에는 콜론(:)이 붙고, 쉼표(,)로 name/value 쌍을 구분한다. array는 value의 순서화된 컬렉션으로, 좌 대괄호([)로 시작하여 우 대괄호(])로 끝낸다. 값들은 쉼표(,)로 구분되며, 큰따옴표(“) 안에 string, number, true, false, null, object, array 등이 올 수 있다. string은 큰따옴표(”) 안에 둘러싸인 유니코드(Unicode) 문자들의 조합이며, 하나의 문자도 string으로 표현된다. number는 8진수와 16진수 형식을 사용하지 않는 것을 제외하면 일반적인 C 계열 언어와 비슷하다[6]-[8].

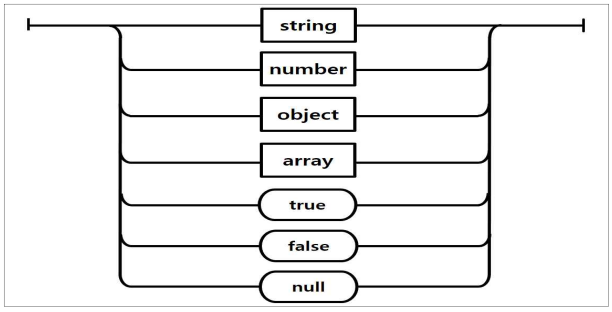

그림 2. JSON Value 형식[9]
Fig. 2. JSON value format

2-3 JSON 라이브러리

1) JsonUtility

Unity Engine 5.3버전부터 엔진 자체에 내장되어, JSON 데이터와 Unity 객체 간의 직렬화 및 역 직렬화를 위한 기본 기능을 제공하고 있다. 경량이며 쉽게 사용할 수 있는 특징이 있지만, 좀 더 복잡한 JSON 데이터 구조에 대해서는 제한이 있다.

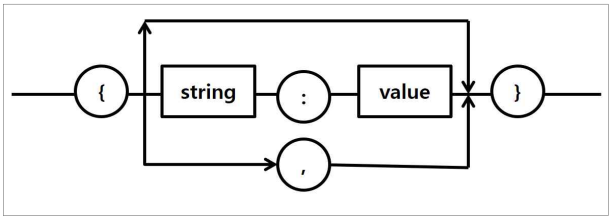

그림 3. JSON 객체 형식[10]
Fig. 3. JSON Object Format

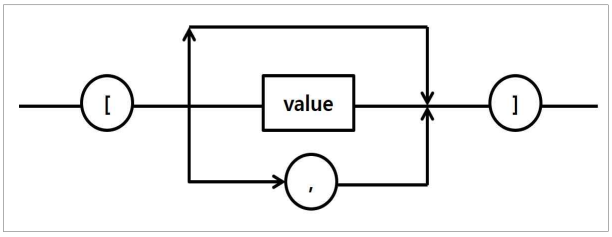

그림 4. JSON 배열 형식[10]
Fig. 4. JSON array format

2) LitJson

Unity Engine 용으로 설계된 인기 있는 JSON 라이브러리 중 하나이며, 간단하고 쉽게 사용할 수 있음에도 가볍고 빠르다. JSON 데이터의 직렬화 및 역 직렬화를 처리할 수 있고, JSON 데이터를 Unity 객체로 변환하거나 반대로 JSON 데이터로 변환하는 기능을 제공한다.

3) Newtonsoft.Json

.NET 기반의 JSON 라이브러리이며, 다양한 플랫폼에서 사용할 수 있고, 많은 기능을 제공하여 인기가 있다. Unity Engine과 함께 사용할 수도 있지만, 일부 추가 구성이 필요할 때도 있다. JSON 데이터를 객체로 직렬화 및 역 직렬화하는데 강력한 기능을 제공하며, Unity Engine의 Mono 프레임워크와 호환성은 뛰어나지만, 때에 따라 약간의 성능 저하가 있을 수 있다.

3) 기타 JSON 라이브러리

JSON 라이브러리는 프로그래밍 언어와 플랫폼에 따라 다양하게 사용되는데, Python에서는 ‘json’ 라이브러리가 내장되어 있어 주로 사용하고, Java에서는 Jackson, Gson을 많이 사용하며, C#에서는 Newtonsoft.Json을 주로 사용한다. 이 외에도 다양한 라이브러리가 있고, 프로젝트의 개발환경과 요구 사항에 따라 일반적으로 적절한 라이브러리를 선택하여 사용한다. 표 1은 JsonUtility, LitJson, Newtonsoft.Json의 특징을 플랫폼, 특징, 성능, 의존성, 라이선스로 구분하여 정리한 내용이다.

2-4 게임 제작에서 JSON 활용

JSON은 데이터를 효율적으로 저장하고 교환하기 위해 일

표 1. JSON 라이브러리 특징
Table 1. JSON library features

JSON library	Item	Specification
JsonUtility	Platform	Unity Game Development
	Features	Specialized for Unity serialization
	Performance	Fast and efficient within Unity environment
	Dependency	Included with Unity Engine
	License	Subject to Unity License
LitJson	Platform	Cross-platform (Independent)
	Features	Supports custom serialization
	Performance	Suitable for small to medium-sized data
	Dependency	Independent library, needs separate inclusion
	License	MIT License
Newtonsoft. Json	Platform	Cross-platform (Independent)
	Features	Rich features and custom support
	Performance	Generally good performance with optimization options
	Dependency	Independent library, needs separate inclusion
	License	MIT License

반적으로 사용되는 데이터 형식으로 인간과 기계 모두에게 읽기 쉽고 이해하기 쉬운 형식의 구조로 되어 있고, 대부분의 프로그래밍 언어에서 쉽게 다룰 수 있는 JSON 라이브러리를 제공하기 때문에 게임 제작 시에 다양한 용도로 유용하게 사용된다. 게임 제작 시에 주로 사용되는 JSON의 용도로는 첫 번째, 게임 데이터 저장 및 로드 하는 용도로 게임의 상태, 플레이어 정보, 아이템, 맵 등의 데이터를 JSON 형식으로 저장하여 게임 실행 시 게임 진행 상태를 저장하고 불러오는 용도로 사용할 수 있다. 두 번째, 게임 설정 관리 용도로 게임의 세부 사항으로 게임 난이도, 음량, 언어 설정 등을 효율적으로 관리하는 용도로 사용할 수 있다. 세 번째, 퀘스트 및 스토리 데이터 관리 용도로 게임 내에서 주어지는 퀘스트, 스토리 이벤트, 대화 등의 데이터를 JSON으로 저장하여, 게임 제작자가 이를 통해 내용을 쉽게 추가하거나 수정할 수 있는 용도로 사용할 수 있다. 네 번째, 다국어 지원 용도로 다국어 문자열을 JSON 파일로 저장하여 각국의 언어별로 텍스트를 효율적으로 관리하여, 언어별로 쉽게 변경하거나, 새로운 언어 추가하는 용도로 사용할 수 있다. 다섯 번째, 애니메이션 및 액션 데이터 제작 용도로 애니메이션, 이동 경로, 플레이어의 행동, 게임 캐릭터의 동작 등을 정의하고 수정하는 용도로 사용할 수 있다. 여섯 번째, UI 제작 용도로 게임의 버튼, 메뉴, 패널 등의 UI(User Interface) 위치나 속성을 JSON 형식으로 저장하여 효율적으로 구축하고 수정할 수 있게 제작하는 용도로 사용할 수 있다. 일곱 번째, 게임 레벨 및 맵 디자인 용도로 게임 레벨 또는 맵의 데이터를 JSON 형식으로 저장하여 이를 통해 레벨을 쉽게 수정하고 새로운 레벨을 추가하는 용도로 사용할 수 있다. 이처럼 다양한 용도로 활용이 가능하며, 표 2는 게임 제작 시 주로 사용되는 JSON의 용도를 정리한 내용이다.

표 2. 게임 제작 시 주로 사용되는 JSON의 용도
Table 2. The purposes of JSON commonly used in game development

Purpose	Description
Game Data Save and Load	Utilizes JSON to store and load game states, player information, items, and map data during runtime, aiding in the permanent storage and retrieval of the game's progress
Game Settings Management	Manages detailed game settings using JSON files, efficiently handling aspects such as game difficulty, volume, language, and allowing for easy modifications
Quest and Story Data	Stores data for quests, story events, and dialogues in JSON format, enabling developers to easily add or modify content when introducing new elements to the game
Localization (Multilingual Support)	Saves multilingual string data in JSON files, facilitating efficient text management for each language. This allows for easy language changes and additions
Animation and Action Data	Utilizes JSON for defining and modifying animations, movement paths, and player actions, providing enhanced flexibility in defining and altering game character movements
User Interface (UI) Management	Stores UI elements (buttons, menus, etc.) in JSON files, streamlining the efficient construction and modification of the game's user interface
Level and Map Design	Manages game level or map data in JSON format, allowing easy modifications and the addition of new levels, simplifying the process of adjusting or expanding game environments

2-4 Unity Engine

최근 게임 및 콘텐츠 개발 분야에서 많이 사용되는 Unity Engine은 초기 주로 게임 제작 용도로 시작했지만, 현재 다양한 분야에서 활용이 늘어나면서 콘텐츠 통합 저작 도구로 크게 성장했다. Unity Engine은 멀티플랫폼을 지원하여 하나의 통합된 작업을 통해 플랫폼 간에 별도의 작업 없이 변환할 수 있으며, 엔진 내부에는 물리 엔진과 라이트 맵과 같은 다양한 기능들이 탑재되어 있고, 에셋스토어(Asset Store)를 통해 필요한 에셋(Assets)을 내려받을 수 있어 빠른 제작이 가능하다[11]-[13].

III. JSON 성능 비교

3-1 실험 환경

실험 환경은 게임 서비스 플랫폼에 따라 PC와 모바일 환경으로 구분하였다. 게임 서비스 점유율을 기준으로 PC에서는 Windows, 모바일에서는 Linux 근간인 IOS, Android 플랫

폼 중 서비스 점유율이 더 높은 Android를 선택하였으며, 기기 성능에 따라 High, Mid, Low로 각 3종씩 구성하였다. PC의 경우는 표 3과 같이, 같은 플랫폼 환경설정을 위하여 Windows 10 Pro에서 CPU와 Memory에 초점을 맞춰 성능 차이를 두었으며, Smart Phone의 경우 국내 사용자가 많은 삼성 제품에 대해 성능에 따라 구분하여 구성하였다.

표 3. PC 실험 환경
Table 3. PC experimental environment

Device	Item	Specification
PC 1 (High)	CPU	Core i7-10700K 3.8GHz
	Number of Core	Octa-Core
	Memory	DDR4 32GB
	OS	Windows 10 Pro
PC 2 (Mid)	CPU	Core i7-6700HQ 2.60GHz
	Number of Core	Quad-Core
	Memory	DDR4 24GB
	OS	Windows 10 Pro
PC 3 (Low)	CPU	Core i7-4790 3.60GHz
	Number of Core	Quad-Core
	Memory	DDR4 16GB
	OS	Windows 10 Pro

표 4. 스마트폰 실험 환경
Table 4. Smart phone experimental environment

Device	Item	Specification
Smart Phone 1 (High)	Model	Galaxy Z Fold 2
	CPU	Qualcomm Snapdragon 865+ SM8250-AB
	Memory	12 GB LPDDR5 SDRAM
	OS	Android 13
	Release date	2020. 9.18
Smart Phone 2 (Mid)	Model	Galaxy S9+
	CPU	Exynos 9 Series (9810)
	Memory	6 GB LPDDR4X SDRAM
	OS	Android 10
	Release date	2018. 3.16
Smart Phone 3 (Low)	Model	Galaxy A30
	CPU	Exynos 7 Series (7904) Octa-Core
	Memory	3/4 GB LPDDR4X SDRAM
	OS	Android 11
	Release date	2019. 5. 3

3-2 실험 내용

JSON 라이브러리의 직렬화, 역 직렬화 속도 측정을 위해 Windows/Android용으로 실험에 사용할 애플리케이션(Application)을 제작하였다. JsonUtility의 경우 Unity Engine에 내장되어 있으므로, 별도의 작업이 필요 없고, LitJson과 Newtonsoft.Json의 경우는 라이브러리를 내려받아 프로젝트에 포함하였다. Data Setting은 JSON 라이브러리에 적재시킬 데이터양을 나타내며, 100개를 시작으로 50000개까지 순차적으로 실험이 진행되도록 구성하였다. 직렬화, 역 직렬화 속도 측정은 C#의 Stopwatch 기능을 활용하여 구간을 측정, 총 10번을 반복한 후 평균값을 산출하여 출력되도록 하였고, 최종 설정된 50000개 데이터양까지 실험이 완료되면 UI에 실험 완료 표시를 한 뒤 애플리케이션이 종료되도록 하였다. 그림 5는 실험 애플리케이션의 알고리즘 흐름도를 나타낸다.

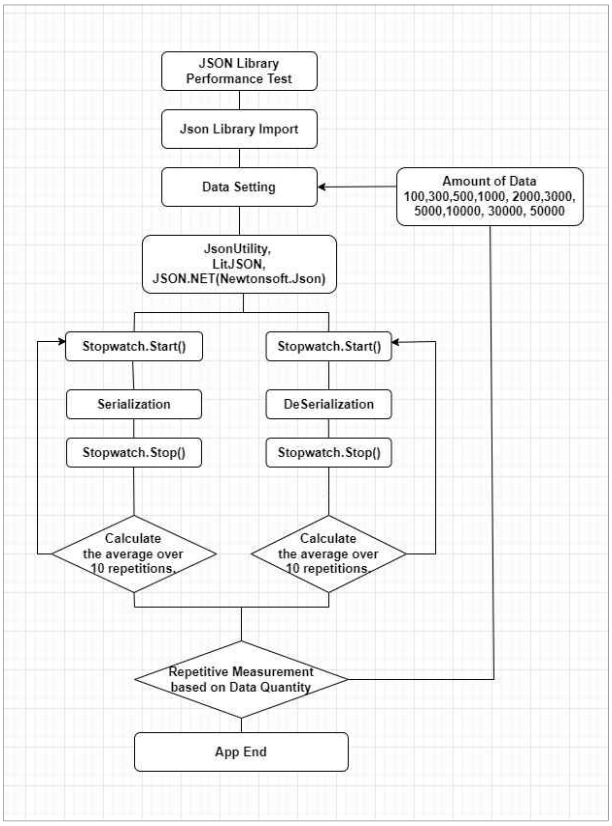

그림 5. 실험용 애플리케이션 알고리즘 흐름도
Fig. 5. Test application algorithm flowchart

```
[Serializable]

public struct TestData
{
    public int level;
    public int hp;
    public int mp;
    public int exp;
}
```

그림 6. 테스트용 데이터 구조체
Fig. 6. Test data structure

Data Setting에서 그림 6과 같이 일반적인 게임에서 많이 사용될 만한 간단한 데이터 구조체를 만들어 실험에서 설정한 데이터양만큼 List 자료 구조를 사용하여 각 JSON 라이브러리에 적재시켰다.

진행 단계를 UI에 표시하기 위해서는 프로세스에 블록이 걸리면 안 되기 때문에, 그림 7과 같이 코루틴(Coroutine) 함수를 사용하여 처리하였고, 각 JSON 라이브러리별 함수를 별도로 작성하여 구조적으로 디버깅에 쉽도록 코드를 작성하였다.

제작 완료 후 실험 애플리케이션을 Windows와 Android용 실행 파일로 만들어, 실험 환경에서 구성한 PC와 Smart Phone 기기에서 실행 후 속도를 측정하였다. 그림 8은 Windows에서 그림 9는 Android에서 속도를 측정하고 있는 애플리케이션 구동 화면이다. 화면 상단에는 측정 진행 단계를 보여주고, 하단에는 설정된 데이터양에 따라 각 JSON 라이브러리별 순차적으로 측정된 직렬화, 역 직렬화 측정값을 화면에 하나씩 표시 처리했다.

```
// Setting the data amount
int[] testCnt = new int[10]
{ 100, 300, 500, 1000, 2000, 3000, 5000, 10000, 30000, 50000 };


IEnumerator TestFunc()
{
    //Repeat according to the set data amount
    for (int i = 0; i < testCnt.Length; i++)
    {
        TestJsonUtilAvg(testCnt[i]);
        yield return null;

        TestLitJsonAvg(testCnt[i]);
        yield return null;

        TestNewtonJsonAvg(testCnt[i]);
        yield return null;
    }
}
```

그림 7. 코루틴으로 작성된 JSON 라이브러리별 함수
Fig. 7. Test data structure

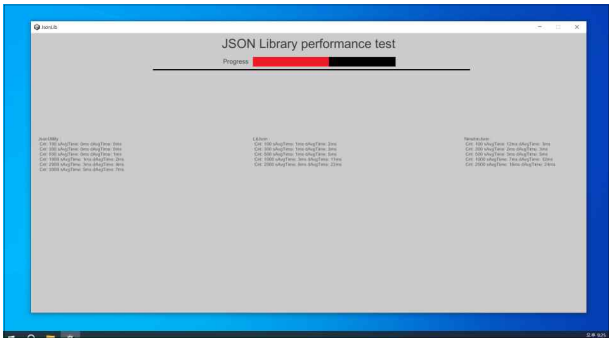

그림 8. Windows에서 성능 테스트
Fig. 8. Performance test on windows

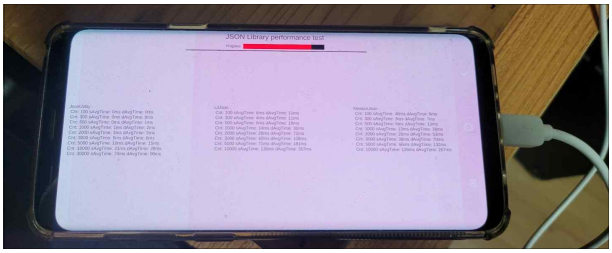

그림 9. Android에서 성능 테스트
Fig. 9. Performance test on android

Ⅳ. 성능 테스트 결과

표 5는 성능에 따른 PC 3종에 대해서 10번 반복 후 직렬화, 역 직렬화 속도의 평균을 측정한 내용이다. Serialization은 'S'로 표기하였고, Deserialization은 'D'로 표기하였다. 기기 성능에 따라서 각 JSON 라이브러리의 성능도 비례하게 측정되었고, 모든 JSON 라이브러리에서 직렬화 속도 보다 역 직렬화 속도에서 더 많은 시간이 소요되었다. 실험한 JSON 라이브러리에서 JsonUtility가 PC 모든 기종에서 4배 정도 빠른 성능을 보였으며, 근소하게 Newtonsoft.Json이 LitJson에 비해 나은 성능을 보였다.

표 5. PC에서 JSON 라이브러리별 직렬화, 역 직렬화 결과
Table 5. Serialization and deserialization results of JSON libraries on PC

Device	Amount of data	JsonUtility		LitJson		NewtonJson	
		S	D	S	D	S	D
PC 1 (High)	100	0	0	1	1	7	1
	300	0	0	0	1	0	1
	500	0	0	1	2	1	1
	1000	0	0	2	5	2	3
	2000	0	1	4	10	4	7
	3000	1	2	6	15	6	10
	5000	1	3	10	26	10	17
	10000	4	7	21	52	20	35
	30000	15	22	63	158	62	104
	50000	26	37	106	263	103	175
Device	Amount of data	JsonUtility		LitJson		NewtonJson	
		S	D	S	D	S	D
PC 2 (Mid)	100	0	0	1	3	12	2
	300	0	0	1	2	0	1
	500	0	0	1	4	1	2
	1000	0	1	3	8	3	5
	2000	1	2	7	17	7	11
	3000	2	3	11	26	10	17
	5000	3	5	18	44	20	31
	10000	7	11	36	89	35	57
	30000	23	35	107	264	107	179
	50000	40	59	183	432	164	284
Device	Amount of data	JsonUtility		LitJson		NewtonJson	
		S	D	S	D	S	D
PC 3 (Low)	100	0	0	4	6	18	5
	300	0	0	2	6	2	3
	500	0	0	5	10	4	7
	1000	1	1	10	23	8	13
	2000	3	4	17	41	17	29
	3000	4	5	27	71	26	45
	5000	7	12	50	121	47	85
	10000	17	29	86	212	78	142
	30000	55	69	253	656	243	435
	50000	84	117	448	1140	457	814

그림 10, 11은 표의 데이터를 기반으로 PC 환경에서 각 JSON 라이브러리의 직렬화, 역 직렬화 성능을 비교한 그래프이다. 세로축은 측정 속도의 단위 millisecond를 나타내며, 가로축은 데이터양을 나타낸다. 그래프에서도 직렬화, 역 직렬화 모두 JsonUtility의 성능이 가장 높음을 알 수 있다.

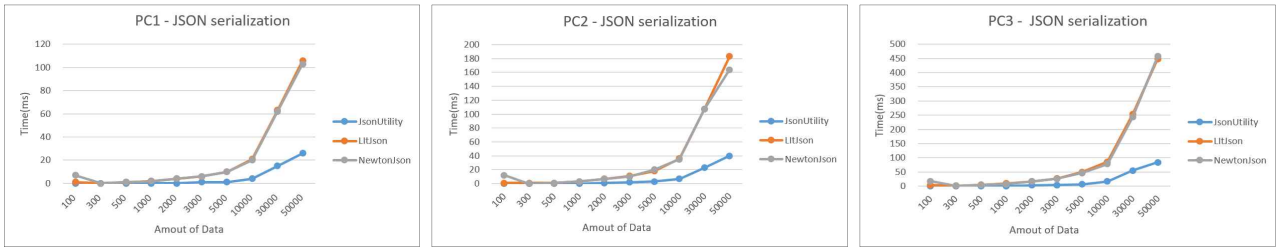

그림 10. PC 기기별 JSON 라이브러리 직렬화 비교
Fig. 10. Comparison of JSON library serialization across PC devices

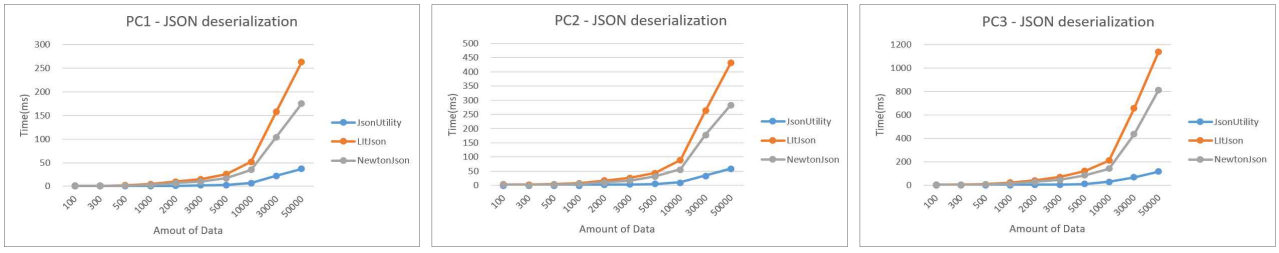

그림 11. PC 기기별 JSON 라이브러리 역 직렬화 비교
Fig. 11. Comparison of JSON library deserialization across PC devices

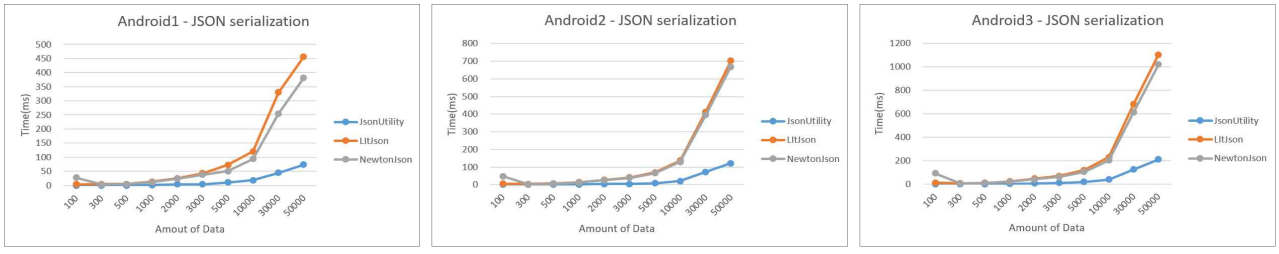

그림 12. Android 기기별 JSON 라이브러리 직렬화 비교
Fig. 12. Comparison of JSON library serialization across android devices

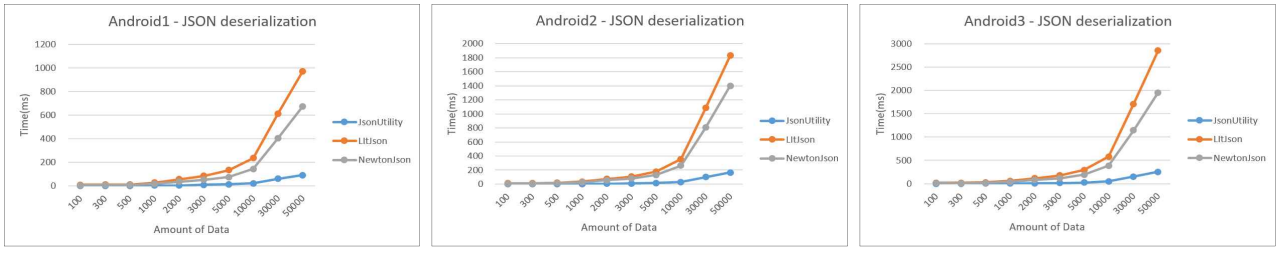

그림 13. Android 기기별 JSON 라이브러리 역 직렬화 비교
Fig. 13. Comparison of JSON library desrialization across android devices

표 6은 Smart Phone에서 측정된 결과이다. PC 환경과 같이 Android 3종에 대하여 10번 반복 후 직렬화, 역 직렬화 속도의 평균을 측정하였다.

휴대용 이동기기 특성상 PC 환경보다는 낮은 성능을 보였다. PC 환경과 마찬가지 기기 성능에 따라 각 JSON 라이브러리의 성능도 비례하게 측정되었고, 모든 JSON 라이브러리에서 직렬화 속도 보다 역 직렬화 속도에서 더 많은 시간이 소요되었다. Smart Phone 환경에서는 JsonUtility가 5~6배 정도 더 나은 성능을 보였고, PC 환경에서처럼 Newtonsoft.Json이 LitJson에 비해 약간의 나은 성능을 보였다. 그림 12, 13은 표의 데이터를 기반으로 비교한 그래프이다.

이처럼 PC와 Smart Phone 환경 모두 JsonUtility가 LitJson과 Newtonsoft.Json 보다 4~6배 정도 빠르다는 결과를 얻었다. 게임 제작 시에 주로 사용되는 JSON의 용도 중에 게임 데이터 저장 및 로드 하는 용도나 퀘스트 및 스토리 데이터 관리 용도와 같이 직렬화, 역 직렬화가 빈번하게 발생하고, 속도가 중요한 용도에는 JsonUtility를 사용하는 것이 효과적이다.

표 6. Smart Phone에서 JSON 라이브러리별 직렬화, 역 직렬화 결과

Table 6. Serialization and deserialization results of JSON libraries on SmartPhone

Device	Amount of data	JsonUtility		LitJson		NewtonJson	
		S	D	S	D	S	D
Smart Phone 1 (High)	100	0	0	4	8	28	5
	300	0	1	5	11	3	7
	500	0	1	5	11	5	7
	1000	1	3	13	28	12	18
	2000	4	3	25	56	24	35
	3000	4	8	43	85	38	52
	5000	11	12	73	133	51	74
	10000	18	21	120	235	94	143
	30000	45	60	330	610	255	402
	50000	74	91	456	972	381	674
Device	Amount of data	JsonUtility		LitJson		NewtonJson	
		S	D	S	D	S	D
Smart Phone 2 (Mid)	100	0	0	6	11	48	9
	300	0	0	4	11	3	7
	500	0	1	7	18	6	13
	1000	1	3	14	35	13	26
	2000	4	6	28	71	27	54
	3000	5	9	41	106	39	78
	5000	9	15	70	179	66	130
	10000	21	30	136	353	130	262
	30000	72	101	413	1083	395	807
	50000	122	166	703	1830	668	1401
Device	Amount of data	JsonUtility		LitJson		NewtonJson	
		S	D	S	D	S	D
Smart Phone 3 (Low)	100	0	0	13	21	93	21
	300	1	1	7	19	6	11
	500	1	3	11	30	12	20
	1000	3	5	24	59	21	38
	2000	8	9	48	118	42	78
	3000	12	15	71	178	63	117
	5000	20	26	119	293	106	192
	10000	41	49	229	576	203	383
	30000	126	152	679	1707	610	1141
	50000	212	253	1101	2851	1018	1950

V. 결 론

게임 산업은 기술의 발전과 함께 지속해서 성장하고 있고, 사용자들은 더욱 현실적이고 흥미로운 경험을 원하기 때문에, 게임 제작은 점점 더 방대하고 복잡한 데이터를 다룰 수밖에 없다. 게임 제작에 있어 데이터 처리와 관리는 매우 중요하며, JSON은 구조화된 데이터를 간결하고 효율적으로 표현하는데 적합한 포맷이기 때문에 게임 제작 시에 많이 활용된다. 하지만, 관련 연구나 자료는 부족하고, JSON 라이브러리의 성능이 비슷할 것으로 판단하여, 익숙하거나 사용하기 쉬운 라이브러리를 선택하는 경우가 많다. 이에 본 연구에서는 게임 제작에 많이 사용되고 있는 Unity Engine에서 게임 데이터 처리와 관리를 위해 주로 사용되고 있는 JSON 라이브러리인 JsonUtility, LitJson 및 Newtonsoft.Json의 성능을 평가하였다. 중요 지표인 직렬화, 역 직렬화 속도에 초점을 맞춰 실험용 애플리케이션을 제작하여 PC와 Smart Phone 플랫폼에서 기기를 성능별로 구분하여 측정하였고, 실험 결과 각 JSON 라이브러리는 실험 환경에서 설정한 기기 성능에 따라 직렬화, 역 직렬화 속도는 비례하게 나타났고, 직렬화 속도 보다 역 직렬화 속도에서 더 많은 시간이 소요되었다. JsonUtility가 다른 라이브러리에 비하여 PC 환경에서는 4배, Smart Phone 환경에서는 5~6배 빠른 성능을 보였으며, 근소하게 Newtonsoft.Json이 LitJson에 비해 나은 성능을 보였다. 하지만 JsonUtility는 기본적인 데이터 타입만 지원하기 때문에 배열을 바로 사용할 수 없어서 JSON 객체를 감싸는 Wrapper 클래스를 따로 만들어서 해결해야 하고, key/value 형태의 값을 저장할 수 있는 자료 구조인 Dictionary는 지원하지 않는 한계가 있다. 반면, Newtonsoft.Json의 경우 데이터 Query 기능을 C#에서 사용할 수 있는 기술인 LINQ(Language- INtegrated Query)를 지원하기 때문에 C#의 배열, 컬렉션, XML, DataSet 등 원하는 데이터만 가져오고 싶은 경우 편리하게 사용할 수 있고, 많은 기능을 제공한다. LitJson의 경우 JsonUtility과 같이 가볍게 사용할 수 있는 장점이 있지만, 무조건 public 변수를 사용해야 하고, float를 지원하지 않는 문제가 있었는데, 최근에 지원하고 있다. 이러한 JSON 라이브러리의 특징과 본 연구 결과를 바탕으로 게임 제작 시 JSON 라이브러리의 선택에 대한 제안은 다음과 같다. 직렬화, 역 직렬화의 속도가 중요한 용도일 때는 JsonUtility를 사용하고, 기본적인 데이터 타입 이외에 가볍게 추가 기능이 필요한 용도로는 LitJson을 사용하며, 더 많은 기능이 필요할 때는 Newtonsoft.Json을 최적화하여 사용하기를 제안한다. 향후 본 연구 결과가 게임 제작 시에 효율적인 참고 자료로 활용되기를 기대해 본다.

참고문헌

[1] ValueCoders. Unreal Engine Vs Unity 3D Games Development: What To Choose? [Internet]. Available: https://www.valuecoders.com/blog/technology-and-apps/unreal-engine-vs-unity-3d-games-development/.

[2] Strabase, Global Game Industry Trend, Korea Creative Content Agency, Naju, p. 40, May-June, 2022.

[3] H. J. Kim, A Study on Data Format for Efficient Game File

I/O, Master's Thesis, Kwangwoon University, Seoul, February 2021.
[4] H. Choi, A Study on Performance Improvement of Web Service and Content Management System (CMS) Using JSON and XML Data, Master's Thesis, Chonnam National University, Gwangju, August 2018.
[5] J. S. Oh and C. G. Song, "Transmission Performance of Improvements in Mobile Applications via XML and JSON Data Translation," in *Proceedings of the Korean Information Science Society Conference*, Vol. 39, No. 1(D), pp. 129-131, June 2012.
[6] H.-S. Kim, N. Han, and S.-J. Lim,"Web Crawler Service Implementation for Information Retrieval Based on Big Data Analysis," *Journal of Digital Contents Society*, Vol. 18, No. 5, pp. 933-942, August 2017. https://doi.org/10.9728/dcs.2017.18.5.933
[7] Tistory. JSON vs XML Parsing [Internet]. Available: http://bolder00.tistory.com/2.
[8] H. J. Kim, *JSON Basic Programming*, Suwon: Humancoding, 2016.
[9] JSON. Introducing JSON [Internet]. Available: https://www.json.org/json-en.html.
[10] TCP School. JSON Basic Structure [Internet]. Available: http://www.tcpschool.com/json/json_basic_structure.
[11] T.-H. Kim, "Content Design and Production for Experiencing Korean, Chinese and Japanese Traditional Masks Using AR Face Recognition Technology," *Journal of Digital Contents Society,* Vol. 23, No. 12, pp. 2355-2364, December 2022. https://doi.org/10.9728/dcs.2022.23.12.2355
[12] T.-E. Kim, "Cat Raising Simulation Using Augmented Reality," Journal of Digital Contents Society, Vol. 24, No. 5, pp. 1131-1138, May 2023. https://doi.org/10.9728/dcs.2023.24.5.1131
[13] Unity Technologies [Internet]. Available: https://unity.com/kr/products/unity-engine

김태환(Tae-Hwan Kim)

2002년 : 용인대학교 전산통계학과 졸업 (이학사)
2019년 : 광운대학교 스마트융합대학원 게임학과 졸업(게임학석사)

2007년～2014년: ㈜핫독스튜디오 개발이사
2015년～2017년: ㈜버드레터 개발이사
2017년～2018년: ㈜스노우픽셀 개발이사
2017년: 한국게임학회 우수논문상 수상
2022년～현 재: 공주대학교 게임디자인학과 박사과정
※관심분야 : 게임, 실감콘텐츠(VR/AR), 융합콘텐츠

경병표(Byung-Pyo Kyung)

1988년 : 영남대학교 응용미술학과
1992년 : 일본 국립큐슈예술공과대학원 예술공학연구과(예술공학석사)
2000년 : 일본 국립큐슈예술공과대학원 예술공학연구과 박사 후기 과정 수료

1996년～현 재: 공주대학교 게임디자인학과 교수
2007년～현 재: 상해공정기술대학교 전가교수
2019년～현 재: 게임물관리위원회 위원
※관심분야 : 게임디자인, 컴퓨터그래픽, 프랙탈디자인