Level Streaming을 활용한 Rogue-Like 게임에서의 Asynchronous Level Load에 관한 연구

이창우[1], 이재호[2], 안혜령[2], 김영종✉
[1]숭실대학교 소프트웨어학부
✉숭실대학교 소프트웨어학부
{cw4932, uss425, ahr0322}@soongsil.ac.kr✉youngjong@ssu.ac.kr

A Study on Asynchronous Level Load in a Rogue-Like Game Using Level Streaming

ChangWoo Lee[1], Jaeho Lee[2], HeaRyeong An[2], Youngjong Kim✉
*School of Software, Soongsil University
✉School of Software, Soongsil University

요 약

Unreal Engine 5의 Level Streaming 기능은 플레이어가 플레이 도중 메모리에 Map을 로드/언 로드하는 등의 작업을 처리하는 기능이다. 이때 Level Streaming은 커다란 Map의 당장 필요한 부분에만 메모리에 로드하고, 렌더링하여 특히 Seamless Open world 장르에서 많이 사용된다. 본 논문에서 제안하는 방식은 이 Level Streaming 기능을 이용하여 현재 개발 진행 중인 Rogue-Like 장르 게임에서 비동기 방식의 로딩 화면과 Stage 전환을 통해 좀 더 부드러운, 즉 더 높은 Frames per Second(fps)를 플레이어에게 제공하기 위한 새로운 스테이지 시스템의 구현방식을 연구한다.

1. 서론

본 저자가 병행으로 Unreal Engine 5의 다양한 기능을 이용하여 제작한 Back-Street는 무작위 Map 생성 기반 Rogue-Like 장르이다. 해당 게임을 제작하는 도중 Unreal Engine 5에서는 Non-Block 방식의 로딩 시스템이 지원하지 않아 본 저자는 초기에 동기-Block의 방식으로 로딩 시스템을 구현하려 했다. 하지만 해당 방법은 제어권을 통째로 Level Load에 넘겨서, 로딩 화면을 멈춰버리는 Block 현상의 원인이 되었다.

이와 같은 현상을 방지하고자 본 논문에서는 Unreal Engine 5에서 제공하는 Level Streaming을 사용하여 Non-Block의 성격을 가진 동기, 비동기 방식을 로딩 화면과 Stage 간의 Level 전환 구현에 적용하였다. 본 논문에서는 두 가지 방식의 차별점이 무엇인지, 상대적 성능(fps)은 어떤 방법이 좋은지 확인하려 한다.

2. 기존 방식

로딩 화면과 Stage 간의 Level 전환 구현에서 기존에 사용해 왔던 동기, Block 방식의 Level System에 대한 이해를 위해 다음 내용을 기술한다. 챕터는 n*n개의 스테이지로 구성되어 있고, Main Menu level에서 Transition Level로 Open Level을 하면, Transition Level의 빈 Level에 로딩 UI를 출력하게 된다. UI를 출력한 후 InGame Level을 Open Level로 호출하게 되는데 이 상황에서 레벨을 생성하는 동안 동기화된 로딩 화면 출력 (Transition Level의 작업인 UI 통한 출력)이 멈춰버리는 현상이 발생한다. 이는 에셋이 완전히 로드될 때까지 다음 단계로 진행하지 않고 대기하여 생기는 문제인데, 이는 게임 경험에 있어서 플레이어에게 화면이 Freezing 된 것 같은 인식을 제공할 우려가 있으며, 프레임 하락의 직접적 원인을 줄 수 있다.

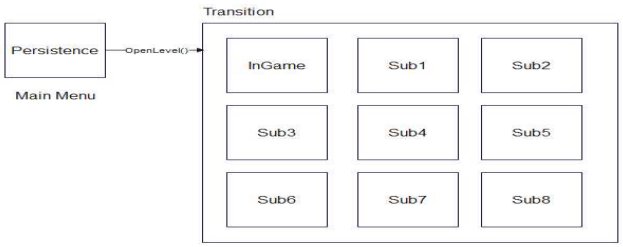
이와는 별개로 구현하였던 방안이 동기-논 블로킹 방식이다. 해당 방식은 Level Streaming을 이용하여 Transition Level에서 Open Level을 하게 되면 빈 Level에서 로딩 UI를 출력하게 되고, 리소스 로드율, 혹은 진행도를 매번 반환받아 로딩 화면에 반영하는 형태는 가능했다. 다만 해당 방식도, 동기식 구조를 기반으로 하기에 많은 리소스를 효율적으로 사용하거나 더 나은 fps를 기대하기는 어려웠다.

3. 제안된 방식 및 실험

3.1 제안된 방식

2.2절에서 확인한 바와 같이 동기식으로 로딩 화면을 구

성하고, Stage System을 생성하는 경우 Block 현상과 리소스를 효율적으로 사용하지 못하는 문제가 있었다. 이에 본 논문에서는 Level Streaming의 비동기 방식을 이용하여 로딩 화면과 Stage System을 구축해 보았다.



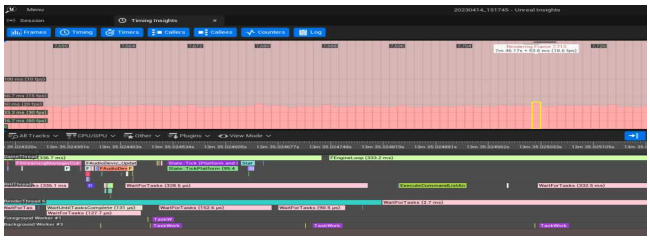
<그림 1> 비동기, Non-Block 방식 Stage System 구성

비동기와 Non-Block 방식의 Stage System은 <그림 1> 을 참고하여 간단히 이해할 수 있다. Main Menu에서 OpenLevel()로 Transition을 부르면 빈 레벨을 생성하고, 해당 레벨에서 로딩 화면을 출력한다. 여기까지는 동기방식과 다름이 없는데, Transition이 다른 레벨들과 종속 관계를 맺으며 InGame 및 Sub 레벨들을 비동기적으로 한 번에 로드 할 수 있는 것이다.

이를 Unreal Engine 5에서 구현하기 위하여 사용되는 것이 Level Streaming 객체이다. 게임플레이가 멈추지 않는 상황에서 비동기 방식으로 Map을 로딩해 온다고 생각하면 좋다. 즉 작업이 병렬적으로 처리되어 게임이 더 부드럽게 작동하며, Level Streaming Priority 기능으로 우선순위를 정하여 게임 성능을 최적화할 때 도움이 된다.

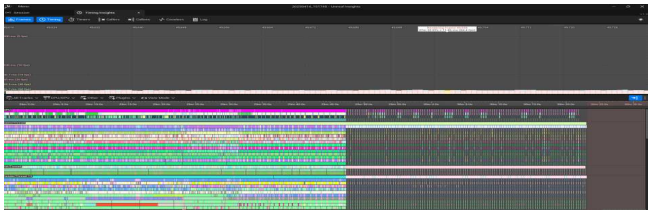
3.2 실험 및 결과

다음은 Level Streaming을 이용하여 실질적인 fps의 영향을 얼마나 끼쳤는지를 확인하기 위해 시스템 통제변인 하에 동기, 비동기 방식의 구분 및 Level Streaming의 장점을 살릴 Load, Unload 기능을 활용하여 명확한 성능 차이를 확인하기 위해 가용할 수 있는 리소스를 모두 로드 한 상태와 동적 로드 했을 때를 조작 변인으로 하여 실험하였다.

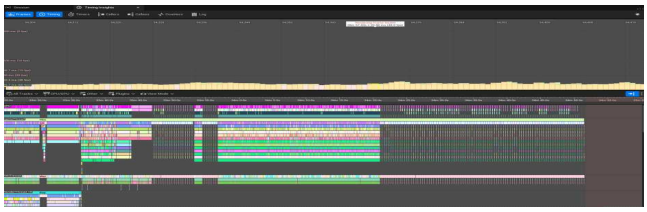


<그림 2> 동기-Non-Blocking 방식을 이용하여 가용할 수 있는 스테이지를 모두 로드 했을 때의 fps

<그림 2>에서는 20fps, <그림3>에서는 40~ 50fps, <그림 4>에서는 80fps로 결과가 나와 Level Streaming 방식을 통해 생성된 Level이 기존 방식에 비해 2배 이상의 성능 효과가 있었으며, Level Streaming의 기능 중 하나인 동적 Load, Unload 기능을 이용해 더 높은 fps를 나타낸다.



<그림 3> Level Streaming을 이용한 비동기-NonBlocking 방식으로 스테이지를 모두 로드 했을 때의 fps



<그림 4> Level Streaming의 비동기-Non-Blocking 방식으로 스테이지를 동적으로 로드 했을 때의 fps

4. 결론

본 논문에서는 Level Streaming을 이용하여 비동기적 Stage 생성 방식을 연구해 보았다. 초기의 예측과 같이 비동기적 생성 방식이 Blocking이 없어 더 매끄러운 게임 경험을 플레이어에게 제공하며, fps도 동기, 블록 방식에 비해 높은 수치를 보여주었다. 이는 Level Streaming이 제공하는 객체의 Load, Unload 기법뿐만 아니라 비동기 방식이 주는 효율적인 리소스 운영이 객관적 수치에 영향을 주었다고 생각된다.

ACKNOWLEDGMENT

"본 연구는 과학기술정보통신부 및 정보통신기획평가원의 SW중심대학사업의 연구결과로 수행되었음"(2018-0-00209)

참고문헌

[1] unreal engine 5, https://docs.unrealengine.com/5.1/ko/level-streaming-overview-in-unreal-engine/

[2] 동기와 비동기, 그리고 블록과 넌 블럭, https://musma.github.io/2019/04/17/blocking-and-synchronous.html

[3] 김용오. "예측 로딩을 통한 온라인게임의 클라이언트 로딩 부하 감소 기법." 국내석사학위논문 고려대학교 컴퓨터과학기술대학원, 2005. 서울