
오픈월드 게임에서
살아있는 세계를 구현하는
기술들
BatStudio Press

오픈월드 게임에서 살아있는 세계(living world)를 구현하는 기술들

ver 1.0

김웅남 지음

BatStudio Press

목차

시작하는 글

최근 가장 주목받는 게임 스타일 중 하나는 단연 **오픈월드**(Open World) 게임입니다. 액션이나 어드벤처는 물론, 롤플레잉 장르까지 다양한 게임들이 오픈월드를 기반으로 제작되고 있으며, 점점 더 많은 플레이어들이 이 방대한 자유도와 몰입감을 즐기고 있습니다.

저는 개인적으로 **트리플 A급 오픈월드 게임 프로젝트** 두 개에 참여한 경험이 있습니다. 이 경험을 통해 오픈월드 게임에서 가장 핵심적인 개념 중 하나인 '살아있는 세계', 즉 **리빙 월드**(Living World)에 큰 관심을 갖게 되었습니다.

당시 저는 프로젝트 팀 내의 리빙 월드 파트와 협업하는 **테크니컬 디자이너**(Technical Designer)로 일하면서, "어떻게 하면 게임 속 세계가 진짜 살아 있는 것처럼 느껴지게 만들 수 있을까?"라는 질문을 직접 부딪히며 탐색할 수 있는 기회를 가졌습니다. 광활하게 펼쳐지는 맵 위에 다양한 생명체와 NPC가 실제로 살아 움직이는 것처럼 보이도록 만들기 위해, 제한된 하드웨어 성능과 한정된 리소스 안에서 어떤 기술이 사용되는지, 어떤 방식으로 설계와 구현이 이뤄지는지를 깊이 있게 경험할 수 있었습니다.

특히 흥미로웠던 점은, 이러한 시스템이 어느 날 갑자기 만들어진 것이 아니라, 수많은 게임 개발자들이 오랜 시간 동안 기술과 노하우를 축적하고, 공유해온 결과물이라는 사실이었습니다. 저는 이 과정을 통해 정말 많은 것들을 배웠고,

언젠가는 이러한 내용을 정리해 **게임 개발 기술에 관심 있는 분들과 나누고 싶다**는 생각을 오래전부터 해왔습니다.

그러나 막상 글을 쓰려니 어디서부터 어떻게 시작해야 할지 몰라, 책을 쓰다 말기를 여러 번 반복했었습니다. 그러던 중, 꼭 거창한 기술서가 아니더라도 **작은 소책자 형태로 핵심 내용을 정리해 전달하는 것도 좋은 방법이 될 수 있겠다**는 생각이 들어, 이번에 이렇게 짧은 책을 쓰게 되었습니다.

이 책은 **프로그래머에게도 도움이 되지만**, 기본적으로는 **테크니컬 디자이너**를 주요 독자로 생각하고 집필했습니다. 테크니컬 디자이너는 게임 기획자와 프로그래머의 중간에 위치한 역할로, 기술적 이해를 바탕으로 게임 기획을 수행하는 직군입니다. 기술을 이해하는 게임 기획자라고 생각하시면 이해가 쉬울 것입니다.

따라서 이 책에서는 **프로그래밍 코드나 알고리즘**은 등장하지 않습니다. 또한 전문적인 세부 기법을 깊이 다루기보다는, 오픈월드 게임 개발에 사용되는 기술들이 **왜 필요하며**, **어떤 원리로 작동하고**, **어떻게 설계되는지**에 초점을 맞추어 설명하고자 합니다. 기술적인 깊이보다는 개념과 구조, 설계 원리를 중심으로 구성되어 있기 때문에, **기술에 대한 감을 잡고 싶은 분들**에게 적합한 내용이 될 것입니다.

이 책을 읽기 위해 **특별한 사전 지식이 요구되지는 않습니다.** 다만 게임 개발에 관심이 있고, 기본적인 프로그래밍 용어나 게임 개발 기법에 대한 이해가 있다면

책의 내용을 훨씬 더 쉽게 따라오실 수 있을 것입니다. 예를 들어, 책에서는 *A* 경로 탐색 알고리즘(A-STAR Pathfinding)* 같은 기술 용어가 등장하긴 하지만, 해당 알고리즘의 구현 원리까지 알아야 하는 것은 아닙니다. 관련 개념을 간략하게만 알고 계셔도 전반적인 내용 파악에는 무리가 없습니다.

그리고 혹시 모르는 용어나 어려운 개념이 나오더라도 걱정하실 필요는 없습니다. 오늘날에는 **생성형** AI(Generative AI)가 매우 보편화되어 있기 때문에, 모르는 단어가 있다면 여러분이 사용하는 AI에게 질문만 해도 금방 개념을 이해할 수 있는 환경이 조성되어 있습니다. 이처럼 과거에 비해 기술에 대한 접근성과 학습 장벽이 크게 낮아졌다는 점은 매우 고무적인 변화입니다.

따라서 여러분이 이미 가지고 있는 다양한 학습 도구를 적극적으로 활용해가며 이 책에서 다루는 내용을 이해하신다면, 프로그래머로서 직접 게임을 개발하거나, 또는 기획자로서 게임 프로그래머들과 협업할 때 기술적인 구조와 작동 원리에 대한 이해를 바탕으로 훨씬 더 원활하게 소통하고, 효율적으로 일할 수 있을 것입니다.

그럼 이제, 오픈월드 게임의 핵심이라 할 수 있는 **리빙 월드의 세계**로 함께 들어가 보겠습니다.

살아 있는 세계(Living World)란 무엇인가

살아 있는 세계(Living World)의 의미

먼저 오픈월드 게임에서 가장 중요한 개념 중 하나인 살아있는 월드, 영어로는 Living World가 무엇인지에 대해 살펴보겠습니다.

오픈월드 게임 장르를 떠올리면 가장 먼저 생각나는 요소는 광활한 맵, 즉 매우 넓은 게임 세계를 자유롭게 탐험할 수 있다는 점일 것입니다. 하지만 단지 맵이 넓다는 이유만으로는 오픈월드 게임이 경쟁력을 갖기는 어렵습니다. 플레이어가 오픈월드 세계에 진정으로 몰입하게 하려면, 그 세계가 스스로의 의지로 움직이는 다양한

생명체로 가득차 있고, 자체적인 원리에 따라 움직이는 듯한 **현실감**을 전달할 수 있어야 합니다.

예를 들어, 플레이어가 특정 지역을 떠나 있어도, 다시 말해 그 장소를 직접 보고 있지 않더라도, 그 지역은 계속 변화하고 있어야 합니다. 그곳에 살고 있는 NPC 즉, 논플레이어 캐릭터들이 각자의 일상을 보내고, 지역의 환경 변화나 시간의 흐름, 그리고 다양한 사건에 따라 자연스럽게 반응하는 모습을 보여줄 수 있어야 합니다. 이러한 느낌을 플레이어에게 전달할 수 있는 것이 바로 *Living World*, 즉 살아있는 월드라는 개념입니다.

그러나 이러한 세계를 실제로 구현하는 일은 생각보다 쉽지 않습니다. 거대한 월드를 게임 안에 구현하려면 방대한 용량의 메모리가 필요하지만, 실제 콘솔 게임기나 PC의 메모리는 한정되어 있습니다. 따라서 이처럼 제한된 메모리 내에서 수많은 NPC와 다양한 게임 오브젝트의 동작을 **성능 저하 없이** 효과적으로 제어해야 하는 기술적인 과제가 존재합니다.

또한, 플레이어의 행동에 따라 NPC나 게임 세계가 **실시간으로 반응**할 수 있도록 체계적으로 설계된 **동적 시스템**도 필요합니다. 그런 의미에서 살아있는 세계(Living World)란 게임 엔진, 메모리 관리, 인공지능(AI), 이벤트 시스템 등 다양한 기술 요소가 유기적으로 결합되어야만 비로소 구현할 수 있는, 복합적인 기술의 총체라 할 수 있습니다.

살아 있는 월드 개념 이해의 중요성

이러한 개념들을 이해하는 것은 오픈월드 게임 개발을 목표로 하는 게임 개발자들에게 매우 중요합니다. 그 이유는 크게 두 가지로 나눌 수 있습니다.

첫째, Living World(살아있는 월드)는 오픈월드 게임에서 **플레이어 경험의 질을 결정짓는 가장 중요한 요소**이기 때문입니다. 이 시스템이 잘 구현되어 있을수록 플레이어는 게임 세계에 더 깊이 빠져들게 되며, 그만큼 게임의 완성도와 재미도 높아지게 됩니다.

둘째, 살아있는 월드를 구현하기 위해 요구되는 **기술적 기법**들은 **게임 전체의 아키텍처 설계 전반에 영향을 미치는 요소**이기 때문입니다. 따라서 이 개념은 게임의 구조와 진행 방식을 설계하는 초기 단계에서부터 깊이 고려되어야 합니다.

이 소책자에서는 바로 이 살아 있는 월드(*Living World*)라는 개념을 중심으로, 오픈월드 게임 개발 과정에서 실제로 활용할 수 있는 대표적인 기술적 기법들을 게임 테크니컬 디자이너의 관점에서 하나하나 구체적으로 살펴보겠습니다. 더불어 이러한 개념들이 어떻게 실현되었는지를 확인할 수 있도록, 실제 게임 사례들도 함께 분석해보겠습니다.

살아있는 월드(living world)의 모습

그럼 이제 좀 더 구체적으로, 살아있는 월드(Living World)가 어떤 모습인지 살펴보겠습니다.

간단히 말하면, 플레이어가 직접 행동하지 않아도 **스스로 움직이고 변화하는 느낌을 주는 환경, 또는 게임 세계**를 말합니다.

예를 들어, 매우 단순한 게임에서는 마을의 NPC(Non-Player Character, 논플레이어 캐릭터)들이 그저 제자리에서 서 있거나, 플레이어가 다가가면 동일한 대사만 반복하는 모습을 자주 볼 수 있습니다. 이는 초기 롤플레잉 게임에서 흔히 나타나는 전형적인 방식입니다.

하지만 살아 있는 세계(*Living World*)에서는 상황이 완전히 달라집니다. 이 세계에서는 NPC들이 각자 **자신만의 일정을 가지고 생활**합니다. 예를 들어, 아침이 되면 NPC들이 집을 나와 농사를 짓거나 상점에서 일을 하고, 해가 지면 술집에 가서 친구를 만나거나 집으로 돌아가 잠을 자는 식으로, 자신에게 주어진 **하루 일과를 따라 움직이는 모습**을 보입니다. 이처럼 NPC들이 마치 진짜 사람처럼 행동하는 방식은 게임 세계에 생명력을 불어넣는 중요한 요소입니다.

또 하나의 중요한 특징은, 플레이어가 특정 지역을 떠나 있어도 그 지역의 **시간은 계속 흐른다는 점**입니다. 즉, 플레이어가 마을에 없더라도, 날씨가 바뀌고 사건이 발생하며, 이에 따라 NPC와 환경이 반응합니다. 예를 들어, 비가 오면 NPC들이

우산을 쓰거나 상점 문을 닫는 식으로 행동이 달라지고, 밤이 되면 몬스터들이 더 활발하게 움직이는 등 **시간과 사건에 따른 반응**이 일어납니다.

다시 말해, 살아 있는 세계(*Living World*)란 플레이어를 중심으로 세계가 정지해 있는 것이 아니라, **게임 세계 자체가 하나의 생명체처럼 유기적으로 움직이는 환경**을 의미합니다. 이러한 시스템 덕분에 플레이어는 자신이 게임 속에 없을 때에도 세계가 계속해서 돌아가고 있다는 인상을 받게 되며, 이는 곧 높은 현실감과 몰입감을 만들어냅니다.

살아있는 세계 구현의 기술적 도전

이번에는 살아 있는 세계(*Living World*)를 게임 안에서 **구현하는 데 필요한 기술적 도전**에 대해 살펴보겠습니다.

사실 살아있는 월드를 실제로 구현하는 일은 생각보다 훨씬 복잡한 작업입니다. 단순히 캐릭터를 많이 배치하거나, 다양한 오브젝트를 맵에 배치하는 것만으로는 충분하지 않으며, 이를 훨씬 뛰어넘는 **정교한 기술적 접근**이 필요합니다.

우선, 게임 속 세계가 커질수록 메모리에 적재해야 하는 데이터의 양도 기하급수적으로 늘어납니다. 따라서 이 많은 데이터를 **어떻게 효율적으로 메모리에 관리할 것인가**가 중요한 문제가 됩니다.

또한 살아있는 월드에서는 수많은 NPC들이 **각자 자신만의 행동 루틴**을 가지고 움직이는 것처럼 보여야 합니다. 그런데 이 모든 NPC의 행동을 매 순간 실시간으로 계산하게 되면, 컴퓨터나 콘솔의 성능이 이를 감당하지 못하고 게임의 **퍼포먼스가 급격히 저하**될 가능성이 있습니다. 따라서 이 문제를 해결하기 위한 **적절한 조정과 최적화**가 반드시 필요합니다.

예를 들어, 초기 오픈월드 게임 중 하나인 《*The Elder Scrolls III: Morrowind*》에서는 NPC들이 날씨나 시간의 변화와 상관없이 항상 같은 위치에 서 있었습니다. 즉, 단지 정해진 대사만 반복하며 플레이어가 접근할 때만 반응하는 방식으로

구현되어 있었습니다. 이런 방식은 게임 세계가 마치 **정지된 공간**, 즉 **정적인 세계**처럼 느껴지게 했고, 우리가 원하는 살아 있는 세계(*Living World*)라는 인상을 주기에는 충분하지 못했습니다.

하지만 그 이후의 후속작들에서는 보다 **현실감 있고 역동적인 세계**를 구현하려는 시도가 이루어졌습니다. 예를 들어 《*The Elder Scrolls V: Skyrim*》이나 록스타 게임즈의 《*Red Dead Redemption 2*》를 플레이해보신 분들은, NPC들이 각자의 일상을 살아가며, 날씨, 시간, 사건 등에 따라 **자연스럽게 행동이 변화**하는 모습을 보신 것을 기억하실 것입니다.

그런데 이처럼 현실감 있는 세계를 만들기 위해서는 게임 안의 캐릭터, 동물, 환경 등 수많은 요소들을 동시에 시뮬레이션해야 하며, 이로 인해 게임이 처리해야 할 **연산량이 폭발적으로 증가**하게 됩니다. 결과적으로 CPU와 메모리에 **막대한 부담**이 발생할 수밖에 없습니다.

따라서 실제 게임을 개발할 때는 모든 요소를 완벽하게 실시간으로 계산하지 않습니다. 대신, 성능을 유지하면서도 현실감은 살릴 수 있도록, **필요한 부분만 정교하게 표현**하고 나머지는 **단순화**하는 방식을 사용합니다.

예를 들어, 플레이어 가까이에 있는 NPC들은 실제로 상세하게 설정된 행동 루틴을 따라 움직이도록 연산하지만, 멀리 떨어져 있는 NPC들은 단지 시간의 흐름에 따른 결과만 반영하는 식으로 간략하게 처리합니다. 또 화면에 보이지 않는 게임

오브젝트의 움직임은 일시적으로 정지시키거나, 실제로는 보여주지 않고 **단순한 수학적 계산**으로 처리하기도 합니다.

따라서 살아있는 세계(*Living World*)에서는 보이는 모든 요소가 **완벽하게 시뮬레이션되고 있는 것은 아닙니다**. 오히려 현실감 있는 세계와 게임 성능 사이의 균형을 맞추기 위한 **수많은 기술적 타협과 설계의 결과물**이라고 볼 수 있습니다.

다시 말해, 플레이어가 보기에 세계가 완전히 살아있는 듯 느껴지더라도, 그 이면에서는 개발자들이 **세심하게 설계한 최적화 기법들**이 조용히 작동하고 있는 것입니다.

이 소책자에서는 오픈월드 게임을 개발하는 과정에서 실제로 많이 사용되고 있는 **다양한 프로그래밍 기술**과 **게임 개발자들의 노하우**들을 다룰 예정입니다.

이때 단순히 이론적인 내용을 나열하는 데 그치지 않고, 실제 개발 현장에서 마주칠 수 있는 문제 상황들을 소개하고, 이를 해결하기 위해 어떤 방식이 사용될 수 있는지, 그리고 기존에 출시된 게임들에서는 어떤 기술적 접근을 통해 이러한 문제들을 해결했는지를 함께 살펴보겠습니다.

다만, 이 책은 **프로그래밍 책이 아니라 테크니컬 디자인**(Technical Design) 책입니다. 따라서 프로그래머가 직접 코드를 어떻게 작성하는지, 또는 특정 시스템을 어떤 방식으로 구현하는지를 구체적으로 다루지는 않습니다. 대신, 이러한

기술들이 **무엇을 위한 것인지**, 그리고 **어떤 목적과 방식으로 활용되는지**에 대해 전반적인 이해를 돕는 데 초점을 맞추고 있습니다.

프로그래머가 아니더라도 내용을 이해하실 수 있도록 가능한 한 다양한 실제 사례를 바탕으로 설명드릴 예정이니, 기술적 개념이 생소하더라도 책을 두 세번 반복해서 읽으신다면, 여기에서 다루는 내용을 자연스럽게 이해하실 수 있을 것입니다.

그럼 이제 이 책에서 다루게 될 **구체적인 내용들**을 간단히 살펴보는 것으로부터 시작해 보겠습니다.

메모리 관리 기법과 레벨 스트리밍(Level Streaming)

첫 번째로 다룰 주제는 **메모리 관리 기법**입니다. 오픈월드 게임은 지형, 건물, NPC, 식생(식물), 날씨 등 방대한 양의 데이터를 실시간으로 처리해야 합니다. 따라서 이처럼 거대한 세계를 효율적으로 관리하기 위해서는 **메모리를 어떻게 분할해서 사용할 것인지**, 그리고 **어떤 데이터를 언제 메모리에 로드(load)하고 언제 제거(unload)할 것인지**에 대한 **정교한 계획**이 필요합니다.

메모리 관리와 관련하여 주로 살펴볼 기술은 **레벨 스트리밍**(Level Streaming)입니다. 이 기법은 플레이어의 위치나 이동 경로에 따라, 필요한 지역만 실시간으로 로딩하고, 더 이상 필요하지 않은 지역은 **메모리에서 제거**하는 방식으로 동작합니다.

이 방식을 사용하면 전체 오픈월드 지형을 한꺼번에 메모리에 올릴 필요가 없어지고, **로딩 화면**(Now Loading) 없이 플레이어가 세계를 **자연스럽게 탐험**할 수 있게 됩니다. 즉, 플레이어가 특정 지역에서 멀어지면 해당 지역의 데이터는 메모리에서 지워지고, 새로운 지역으로 이동할 때는 그에 맞는 데이터가 자연스럽게 로드되는 방식이라고 이해하시면 됩니다.

이와 같은 레벨 스트리밍(Level Streaming) 방식을 통해 플레이어는 몰입을 방해하는 끊김 현상에서 벗어나, **연속적인 게임 플레이 경험**을 할 수 있게 됩니다.

월드 분할(World Partition)

다음으로는 **월드 분할**(World Partition)이라고 불리는 기법을 살펴보겠습니다. 이 기술은 **하나의 거대한 게임 세계를 여러 개의 작은 구역으로 나누어 관리**하는 것을 말합니다.

이렇게 큰 지역을 여러 개의 작은 구역 단위로 나누면, 각 구역의 데이터를 **독립적으로 로딩**(loading)**하거나 언로딩**(unloading)할 수 있습니다. 즉, 플레이어가 접근한 구역만 메모리에 올리고, 필요하지 않은 구역은 메모리에서 제거함으로써, 전체 월드를 동시에 처리하지 않아도 되도록 최적화할 수 있는 것입니다.

LOD(Level of Detail) 시스템

그 다음으로 살펴볼 주제는 LOD(Level of Detail) 시스템입니다. 이 기술은 주로 **그래픽 성능을 최적화**하기 위해 사용됩니다. 핵심 개념은 **플레이어와의 거리**에 따라 게임 오브젝트의 **디테일 수준**을 조절하는 것입니다.

즉, 플레이어로부터 멀리 떨어진 오브젝트는 단순하고 심플한 형태로 렌더링하고, 플레이어에 근접한 오브젝트는 고해상도의 정교한 모델로 렌더링하는 방식입니다. 예를 들어, 멀리 있는 산은 단순한 삼각형 형태로 표현되지만, 플레이어가 그 산에 가까이 가면 점점 더 **디테일한 고해상도 모델**로 교체됩니다.

이러한 방식은 **시각적 품질은 유지하면서도 프레임 속도는 안정적으로 확보**할 수 있게 해주므로, 오픈월드 게임을 비롯한 대부분의 3D 게임에서 널리 사용되는 중요한 기술입니다.

참고로 오픈 월드 게임에서 LOD 는 그래픽 요소에 대해서만 적용되는 기법만은 아닙니다. 적 AI 처리를 위한 연산에 대해서도 LOD 가 널리 사용됩니다. 따라서 인공지능 LOD 에 대해서도 다루어 볼 예정입니다.

NPC 스케줄링(NPC Scheduling)

다음으로 살펴볼 주제는 NPC 스케줄링(NPC Scheduling)입니다. 이 기술은 게임 내의 NPC가 **시간대나 특정 상황에 따라 어떤 행동을 할지 결정하는 시스템**입니다.

예를 들어, NPC가 낮에는 일터로 나가서 일을 하고, 밤에는 집으로 돌아가 잠을 자는 식의 **일상적인 루틴을 자동으로 수행**하도록 설계하는 것이 NPC 스케줄링의 중요한 기능입니다. 이처럼 NPC들이 규칙적인 하루를 보내는 것처럼 보이면, 게임 세계는 훨씬 더 **사실적이고 생동감 있게** 느껴지게 됩니다.

다만, 모든 NPC의 스케쥴을 동일하게 처리할 수는 없습니다. **플레이어로부터 가까이에 있는 NPC**, 즉 직접 시야에 보이는 NPC들은 실제 행동을 세밀하게 계산해주는 반면, **플레이어가 보지 않는 위치에 있는 NPC**들은 보다 단순화된 방식으로 루틴을 처리해야 성능을 효율적으로 유지할 수 있습니다.

이벤트 기반 반응 시스템(Event-Driven System)

마지막으로는 이벤트 기반 반응 시스템(Event-Driven System)에 대해 알아볼 예정입니다. 이 시스템은 게임 내에서 발생하는 다양한 사건(이벤트)이나 **플레이어의 행동(Action)**에 따라 게임 세계가 **자동으로 반응**하도록 설계하는 것을 말합니다.

예를 들어, 플레이어가 마을에서 싸움을 벌이면 주변의 주민들이 도망가고, 이를 본 **경비병** NPC가 반응하여 출동하는 등의 상황이 발생합니다. 이처럼 특정 이벤트가 발생했을 때, 이에 대응하는 NPC**의 행동이나 환경 변화**가 자동으로 작동하도록 고안된 시스템입니다.

이러한 이벤트 기반 시스템이 **어떻게 구축되는지**, 그리고 다양한 게임들에서 **어떤 방식으로 사용되어 왔는지**에 대해서도 앞으로 살펴볼 예정입니다.

책 구매 안내

책의 미리보기 내용이 마음에 드셨나요?

책의 나머지 부분을 더 읽어 보시고 싶다면 다음의 정식판 사이트를 방문해서 구매하시기 바랍니다.

https://batstudio.gumroad.com/l/dgzzkb