게임 엔진과 절차적 콘텐츠 생성(PCG) 알고리즘을 활용한 가상 환경 구축 및 렌더링 최적화

최승민*, 양영주*, 권혁진*, 김정훈*, 황준상*, 김민철*
*서일대학교 AI게임융합학과
seungmin7954@gmail.com, yangyeongjoo1004@gmail.com,
yujhee97@gmail.com, kjh0403021@gmail.com, gimnora01@gmail.com,
take@seoil.ac.kr

Construction of Virtual Environments and Rendering Optimization Using Game Engines and Procedural Content Generation (PCG) Algorithms

Seung-Min Choi*, Young-Ju Yang*, Hyuk-Jin Kwen*, Jung-Hun Kim*, Jun-Sang Hwang*, Min Chul Kim*
*Dept. of AI Game Convergence, Seoil University

요 약

본 논문에서는 유니티(Unity) 게임 엔진을 활용하여 대규모 가상 환경(디지털 트윈)을 구축할 때 발생하는 연산 병목과 메모리 누수 문제를 해결하기 위한 데이터 중심 아키텍처(Data-Oriented Architecture)를 제안한다. 기존 시뮬레이터의 한계를 극복하기 위해 게임 개발 분야에서 널리 쓰이는 절차적 환경 생성(PCG) 기법을 도입하였으며, 특히 노이즈 합성과 템플릿 기반 구조 확장 기법을 결합하여 다양한 지형과 객체 분포를 효율적으로 생성하였다. 본 연구는 제안한 아키텍처의 성능 검증을 위해 물류망 테스트베드를 사례로 적용하였으며, 수만 개의 물리적 객체 연산 부하를 줄이기 위해 플라이웨이트 패턴 및 비동기 렌더링 등 게임 클라이언트 최적화 기술을 융합하였다. 검증 결과, 기존 방식 대비 약 11배 규모(102,400개 타일)의 대규모 환경에서도 Batches 호출 횟수를 5회 미만으로 단축하고 120 FPS 이상의 안정적인 프레임레이트를 확보하였다. 본 연구는 향후 고도화된 시뮬레이션 시스템을 위한 고성능 가상 인프라 및 대규모 시뮬레이션 게임의 코어 엔진 설계로 활용될 수 있다.

1. 서론

현대의 디지털 트윈(Digital Twin) 및 가상 현실 기술이 고도화됨에 따라, 수만 개 이상의 동적 엔티티와 대규모 데이터가 상호작용하는 대규모 가상 환경을 실시간으로 시뮬레이션하는 기술의 수요가 급증하고 있다[9]. 그러나 기존의 객체 지향 렌더링(Object-Oriented Rendering) 방식을 활용한 시뮬레이터는 객체 수가 증가함에 따라 기하급수적인 연산 병목과 프레임 지연을 유발한다. 또한, 고도화된 인공지능 등을 테스트하기 위해서는 다양한 지형적 변수가 존재하는 대규모 환경이 필요하나, 이를 수동으로 구축하는 데는 막대한 비용이 소모된다[7, 8].

기존의 산업용 시뮬레이터는 정밀한 수치 해석에 치중하여 수만 개 이상의 동적 객체를 실시간으로 렌더링하고 상호작용하는 데 물리적인 한계를 지닌다. 반면, 최신 범용 게임 엔진(Unity)은 데이터 지향 설계(Data-Oriented Design)와 고속 렌더링 파이프라인(Rendering Pipeline)을 통해 거대한 규모의 객체를 지연 없이 처리하는 아키텍처에 특화되어 있다. 이에 본 연구는 게임 클라이언트의 핵심 엔진 프로그래밍 기술(메모리 경량화, 배치 렌더링 최적화)을 시뮬레이션에 융합하여 시스템의 연산 병목을 극복하는 최적화 인프라를 설계한다. 나아가 노이즈 합성 및 템플릿 확장 기반의 절차적 생성 알고리즘을 응용하여 대규모 가상 환경을 구축하고, 이를 대규모 가상 환경 아키텍처에 적용하여 성능 검증을 수행하는 새로운 접근법을 제시하고자 한다.

2. 고성능 데이터 인프라 아키텍처

2.1 플라이웨이트 패턴 및 가상 그리드 적용

초대규모 시뮬레이션을 지연 없이 구동하기 위해, 물리적 렌더링과 논리적 데이터 연산을 철저히 분리하는 아키텍처를 적용하였다. 수만 개의 타일 데이터를 개별 인스턴스로 할당할 경우 발생하는 가비지 컬렉터(GC) 부하를 제거하기 위해 플라이웨이트 패턴(Flyweight Pattern)을 적용하였다. 시뮬레이션 공간은 물리적 좌표계가 아닌 순수 정수형 좌표계 기반의 2차원 가상 그리드(Virtual Grid)로 추상화하였으며, 타일의 고유 상태값만을 가벼운 구조체(Struct)로 관리하여 메모리 점유율을 극소화하였다.

2.2 비동기 시분할 연산 및 버퍼 일괄 적용

기존 방식에서는 시뮬레이션 객체가 생성될 때마다 메인 스레드에서 그래픽 API를 호출하여 심각한 프레임 드랍이 발생하였다. 이를 해결하기 위해 환경 데이터 연산 로직과 시각화 층을 완전히 분리하는 데이터-뷰 분리(Data-View Separation) 원칙을 도입하였다. 코루틴(Coroutine)을 활용하여 대규모 데이터 배열 연산을 시분할 비동기 처리(Time-Slicing Asynchronous)로 분산시켰으며, 렌더링 시에는 1차원 컬러 버퍼 배열에 계산된 픽셀 데이터를 적재한 후 텍스처에 단 한 번의 그래픽 호출로 덮어씌우는 일괄 스왑 방식을 채택하여 O(1) 수준의 시각화 부하를 달성하였다[5].

[그림 1] 시스템 아키텍처 다이어그램

3. 절차적 환경 생성 알고리즘

구축된 고성능 데이터 인프라를 바탕으로, 시스템 동작중 발생할 수 있는 엣지 케이스(Edge Case)를 시뮬레이션하기 위해 다중 노이즈 알고리즘 기반의 환경 생성기를 도입하였다[2, 6]. 단순 난수가 아닌, 파라미터 제어를 통해 수학적으로 통제 가능한 지형 데이터를 합성(Synthetic Data Generation)한다.

3.1 FBM 및 심플렉스 노이즈 기반 지형 생성

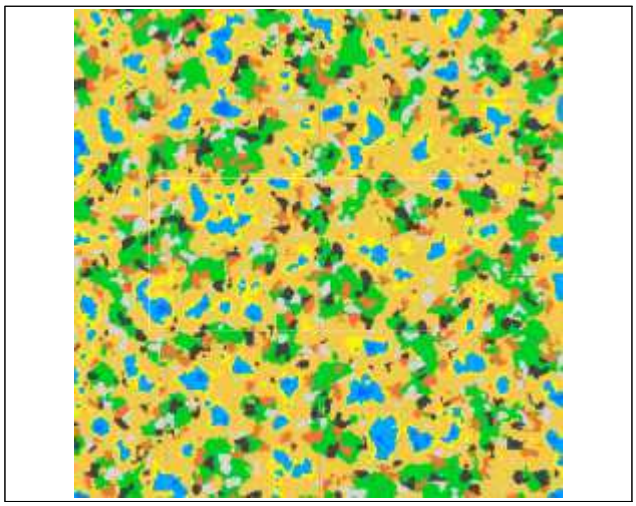
기존 펄린 노이즈의 방향성 아티팩트가 개선된 심플렉스 노이즈(Simplex Noise)에 FBM(Fractal Brownian Motion) 기법을 결합하였다. 주파수, 진폭, 옥타브 파라미터 중첩 연산을 통해 연속성을 지닌 자연스러운 대륙과 대규모 가상 지형을 절차적으로 생성하였다[2, 3]. 본 연구에서 적용한 FBM(Fractal Brownian Motion) 기법의 옥타브 중첩 연산은 다음과 같은 수식으로 정의된다.

$$H(x,y) = \sum_{i=0}^{n-1} a_i \cdot noise(x \cdot f_i, y \cdot f_i)$$

여기서 n은 옥타브 수, a_i는 진폭(Amplitude), f_i는 주파수(Frequency)를 의미한다.

3.2 셀룰러 노이즈와 다중 자원 분배 알고리즘

자원이 밀집된 광맥 패턴을 생성하기 위해 특정 시드 중심점으로부터의 거리를 계산하는 셀룰러 노이즈(Cellular Noise)를 활용하고[5], 지형의 굴곡에 따라 가장자리가 불규칙하게 침식되도록 노이즈 곱셈 마스킹(Noise Multiplication Masking)을 적용하였다. 다수의 자원이 특정 좌표에서 중첩될 때 발생하는 혼합 현상을 제어하기 위해 각 자원의 산출값을 도출하고, 설정된 임계값을 초과하는 최댓값만을 배치하는 승자 독식 판별 알고리즘을 도입하여 명확한 자원 및 엔티티 군락 경계를 구현하였다.



[그림 2] 16 x 16 크기를 가진 20 x 20 청크의 절차적 지형 생성 결과

4. 실험 및 결과

[표 1] 최적화 기법 적용 전후 렌더링 성능 비교표

	렌더링 객체 수	Batches 호출 횟수	FPS
GameObject 인스턴스화	9,216개 6 x 6 청크	10,000회	10 이하
데이터 버퍼 일괄 렌더링	102,400개 20 x 20 청크	5회 미만	120 이상

제안된 아키텍처의 성능 검증을 위해 렌더링 프로파일러를 활용하여 정량적 지표를 비교 분석하였다. 기존의 개별 객체 인스턴스화 방식은 타일 수가 9,216개 수준부터 심한 한계 프레임 드랍을 보였으나, 제안한 비동기 버퍼 일괄 적용 및 플라이웨이트 패턴을 적용한 결과 타일 수가 102,400개를 초과하는 극단적인 대규모 확장(기존 9,216개 대비 약 11배 규모)에서도 Batches 호출 횟수가 5회 미만 수준으로 급감하였다.

시뮬레이션 프레임레이트(FPS) 또한 120 이상으로 안정적으로 방어됨을 확인하였으며, 연산 중에는 백그라운드 데이터 처리가 완료될 때까지 독립적인 스레드 흐름을 유지하여 메인 렌더링 스레드의 프리징(Freezing) 및 연산 병목 현상을 원천 차단하였다.

5. 결론 및 향후 연구

본 연구는 대규모 가상 환경 시뮬레이터 구축 시 발생하는 연산 병목을 극복하기 위해, 데이터 중심 아키텍처 기반의 최적화 인프라와 절차적 환경 생성 알고리즘을 융합하여 설계하고 물류망 테스트베드 적용 사례를 통해 그 유효성을 정량적으로 검증하였다. 향후 연구에서는 본 시스템 상에 통합 ID 체계와 고정 틱(Tick) 기반의 제어 알고리즘을 도입하여, 시뮬레이션 내부의 객체 상호작용 및 논리적 흐름을 고도화할 계획이다[10].

본 연구에서 구축한 데이터-뷰 분리 아키텍처와 절차적 생성 기법은 향후 유니티 엔진의 차세대 기술인 DOTS(Data-Oriented Technology Stack) 및 ECS(Entity Component System) 구조와 결합하여 연산 효율을 한 차원 더 끌어올릴 수 있다[1, 4]. 이는 단순한 산업 시뮬레이터를 넘어, 실제 수십만 개의 독립적인 엔티티(Entity)가 실시간으로 상호작용하는 대규모 시뮬레이션 게임(Factory Automation Game)의 코어 엔진 설계로 직접 응용될 수 있다는 점에서 게임 공학과 산업 공학 모두에 실무적인 기여를 한다.

참고문헌

[1] 성만규, "게임 개발을 위한 엔티티-컴포넌트 시스템 프레임워크 성능 비교 분석", 한국멀티미디어학회논문지, 제 28권 1호, pp. 58-65, 2025년.

[2] 원민철, 김완규, 이승교, 이부형, "노이즈 합성을 통한 게임 맵 생성", 한국정보기술학회 추계 종합학술대회 논문집, pp. 1689-1691, 2024년.

[3] 정혜리, 시종욱, 김성영, "시각적 변화 및 동적 전투 영역 기반 절차적 맵 생성 기법: 로그라이크 사례 연구", 한국정보기술학회 추계 종합학술대회 논문집, 2025년.

[4] 최예찬, 이바다, 송창근, 이정, "Unity DOTS의 성능 및 효용성 분석", 한국HCI학회 학술대회(Proceedings of HCIK), 2023년.

[5] 디파인시스템 주식회사, "Unity3D에서 CPU/GPU 부하분산 최적화 기법", 공개특허 10-2024-0047834, 2024년.

[6] 김재현, 이부형, "로그 라이크 게임에서 새로운 맵 생성 알고리즘", 한국정보기술학회논문지, 제 19권 1호, pp. 139-146, 2021년.

[7] 유병화, 하태관, 박태화, 백인창, 김경중, "파동함수 붕괴 알고리즘을 이용한 소형 콘셉트 맵으로부터의 대규모 게임 맵 자동생성", 한국정보과학회 컴퓨팅의 실제 논문지, 2023년.

[8] 이재현, "가변 블록 기반 파동함수 붕괴 알고리즘을 이용한 대규모 맵의 고속 생성", 한국멀티미디어학회논문지, 제 26권 8호, pp. 1253-1261, 2023년.

[9] 박재현, 김재헌, 이채린, 이유미, "조경설계의 친환경성 평가를 위한 디지털 트윈 시뮬레이터 개발과 사용성 분석", 한국조경학회지, 제 53권 6호, 2025년.

[10] 진영훈, 이승화, 김혜경, "포인트 클라우드를 이용한 충돌회피 알고리즘", 사물인터넷융복합논문지, 제 10권 6호, pp. 7-14, 2024년.