JDCS
디지털콘텐츠학회논문지
Journal of Digital Contents Society
Vol. 26, No. 1, pp. 237-245, Jan. 2025

Check for updates

강화학습을 활용한 물체의 파괴 시뮬레이션 개발

문 철 호[1] · 우 탁[2*]
[1]경희대학교 소프트웨어융합학과 학사과정
[2]경희대학교 일반대학원 메타버스학과 교수

Development of Object Fracture Simulation Using Reinforcement Learning

Cheol-Ho Mun[1] · Tack Woo[2*]

[1]Bachelor's Course, Department of Software Convergence, Kyung Hee University, Yougin 17104, Korea
[2]Professor, Department of Metaverse, Graduate School, Kyung Hee University, Yongin 17104, Korea

[요 약]

물체의 파괴 시뮬레이션은 게임의 장르를 가리지 않고 어디서나 활용되는 기술 중 하나이며, 3D 게임 개발의 핵심 요소 중 하나로 꼽힌다. 물체의 파괴 시뮬레이션은 복잡한 연산을 필요로 하게 된다. 따라서 게임에 현실적인 물체 파괴 시뮬레이션을 적용하는 것은 게임의 성능에 큰 영향을 미치게 된다. 이러한 이유 때문에 많은 3D 게임에서는 높은 사양의 컴퓨터 자원들을 요구하거나, 물체에 가해지는 힘이나 방향 등이 적용되지 못하는 단순한 계산의 파괴 시뮬레이션이 적용된다.

본 연구는 다음과 같은 문제를 강화학습을 통해 해결하고자 한다. 복잡한 연산으로 인해 구현이 어려운 점을 해결하고, 물체의 파괴를 구현하는 데에 있어서 더 넓은 표현을 할 수 있도록 하는 것에 목적을 둔다. 강화학습을 통해서 개발자는 Hyperparameter들을 조절하여 원하는 형태의 파괴를 직접 학습시킬 수 있다. 단순히 현실적인 파괴 연출에서 벗어나, 다양한 파괴 시뮬레이션을 게임에 적용할 수 있다.

[Abstract]

Object destruction simulation, a versatile technology used across various game genres, is considered a core element in three-dimensional game development. Object destruction simulation requires complex calculations, greatly impacting the performance when applied to realistic destruction simulations in games. Therefore, many 3D games either demand high-end computing resources or employ simplified destruction simulations that lack realistic forces or directions when external power applied to objects.

This research uses reinforcement learning to alleviate the difficulty of implementing complex calculations and enable a broader range of expressions in object destruction. Through reinforcement learning, developers can adjust hyperparameters to train the desired forms of destruction directly. This approach enables for the application of diverse destruction simulations in 3D games, going beyond simple realistic destruction effects.

색인어 : 강화학습, 파괴 시뮬레이션, 인공지능, 디지털 게임, 게임 공학
Keyword : Reinforcement Learning, Fracture Simulation, AI, Digital Game, Game Engineering

http://dx.doi.org/10.9728/dcs.2025.26.1.237

Received 15 November 2024; **Revised** 02 December 2024
Accepted 14 January 2025

***Corresponding Author; Tack Woo**

Tel: +82-31-201-3890
E-mail: twoo@khu.ac.kr

Ⅰ. 서 론

1-1 연구 배경 및 목적

물체의 파괴 시뮬레이션은 컴퓨터 그래픽스와 물리 기반 연산에서 중요한 주제 중 하나로, 사실적인 파괴 연출을 구현하기 위해 복잡한 연산과 높은 계산 능력이 요구된다. 특히, 자연스러운 파괴 연출을 위해서는 물체의 구조적 특성과 충격, 중력 등 다양한 요소를 고려해야 하는데, 이는 물리 법칙을 기반으로 하는 정교한 시뮬레이션을 필요로 한다.

기존의 연구들은 시뮬레이션의 연산을 정확하게, 그리고 더 빠르게 수행하는 방향으로 진행되어 왔다[1]. 정확하고 빠르게 적용되는 물리 시뮬레이션은 플레이어로 하여금 현실감을 느끼게 만들고, 게임에 몰입할 수 있도록 돕는다. 따라서 그러한 연구는 게임 콘텐츠 개발에 큰 영향을 미칠 수 있다.

그러나 이러한 시뮬레이션은 개발자의 의도와 연출을 반영하여 특정한 방식으로 파괴가 일어나도록 제어하는 것이 어렵다는 문제가 존재한다[2]. 이 연구에서는 강화학습(Reinforcement Learning)을 활용하여 이러한 문제를 해결하고자 한다. 강화학습은 에이전트가 환경과 상호작용하면서 보상을 최대화 하는 방향으로 학습하는 방식으로, 이 특성을 이용해 파괴 연출의 복잡한 요소들을 학습하고 제어할 수 있다. 특히, 개발자가 직접 설정한 하이퍼파라미터(Hyperparameter)를 통해 파괴 시뮬레이션의 다양한 변수들을 조작함으로써, 연출 의도에 맞는 파괴 양상을 구현할 수 있는 가능성을 제시한다. 이를 통해 현실적이고도 제어 가능한 파괴 시뮬레이션을 구현할 수 있으며, 다양한 게임 시나리오에서 응용될 수 있다[3].

1-2 연구 방법

본 연구에서는 Unity와 ML-Agents를 이용하여 물체의 파괴 시뮬레이션을 강화학습 기반으로 구현하는 방법을 제안한다. Unity는 물리 기반 시뮬레이션과 3D 그래픽스 구현을 지원하는 게임 엔진으로, 게임 속 환경 구성에 적합하다. ML-Agents는 Unity에서 제공하는 강력한 강화학습 라이브러리로, 에이전트가 Unity로 구성된 환경에서 학습하며 행동을 최적화할 수 있도록 지원한며, Unity를 사용해 파괴 가능한 물체와 상호작용할 수 있는 3D 환경을 구성한다. 만들어진 학습 환경 내에서 ML-Agents를 통해 에이전트가 학습할 수 있도록 설계하고, 학습 과정에서 hyperparameter를 설정하여 연출을 조작할 수 있도록 한다.

에이전트는 각 에피소드에서 기본 Mesh에 추가적인 정점을 생성하고 그 결과에 따라 보상을 받는다. 학습의 주요 목표는 충격지점에 맞추어 최적의 위치에 정점을 생성하는 것이다. 새로 생성된 정점들의 정보를 기존 정점 배열에 업데이트한 뒤, 메쉬 생성 알고리즘에 따라 여러 조각의 Static Mesh로 분리되도록 구현한다.

Ⅱ. 관련 이론 및 선행 연구

2-1 강화학습(Reinforcement Learning)

강화학습(Reinforcement Learnig, RL)은 에이전트가 환경과 상호작용하면서 보상을 극대화하는 행동을 학습하는 기계 학습의 한 분야이다[4]. 기계 학습의 다른 분야인 지도 학습과는 다르게, 명확한 정답 데이터가 주어지지 않고 에이전트가 스스로 행동을 통해 얻은 경험을 기반으로 최적의 정책을 학습한다는 점에서 차별화된다[5].

강화학습의 학습 과정은 먼저 행동의 주체를 의미하는 에이전트(Agent)가 환경(Environment)와 상호작용하며 학습을 진행한다[6]. 환경에 상태(State)에 따라 에이전트가 어떤 행동(Action)을 취할지 결정한다. 이 행동은 환경의 상태를 다시 변화시키고, 변화된 상태에 따라 행동을 반복하게 된다. 취한 행동에 따라 에이전트는 보상(Reward)를 받게 된다.

이 때, 어떤 행동을 취해야 최대의 보상을 얻을지 결정하는 규칙 및 함수를 정책(Policy)라고 부른다. 강화학습의 궁극적인 목표는 최적의 정책을 학습하는 것이다. 최적의 정책이란, 에이전트의 행동으로 얻는 장기적인 누적 보상을 최대화하는 정책을 의미한다[7].

표 1. 강화학습 기본 구성 요소
Table 1. Basic components of reinforcement learning

Component	Description
Agent	The subject of learning
Environment	The environment where learning takes place
State	The state of the environment
Action	The agent's action based on the state
Reward	The reward given based on the action

2-2 ML-Agents Toolkit

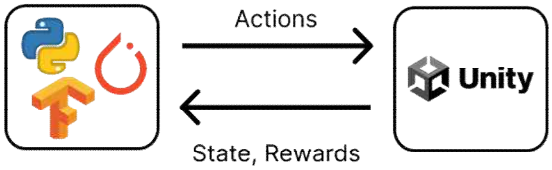

그림 1. Unity ML-Agents의 역할
Fig. 1. Role of Unity ML-Agents

Unity ML-Agents는 Unity 게임 엔진을 사용하여 인공지능 에이전트를 학습시키기 위한 도구이다. 강화학습 및 기타 기계학습 기법을 게임과 시뮬레이션 환경에서 쉽게 적용할 수 있도록 지원하는 학습 툴킷이다[8]. ML-Agents를 통해

Unity에서 학습을 위한 가상 환경을 구축하고, 에이전트가 환경과 상호작용하면서 행동을 학습하도록 설정할 수 있다. 그림 1에서와 같이 ML-Agents는 Python API들과 함께 동작하며, 다양한 강화학습 알고리즘을 활용할 수 있어 복잡한 AI 시스템 개발에 적합하다. 이를 통해 게임 개발, 자율주행, 로봇공학 등의 분야에서 AI 연구를 효율적으로 수행하는 것이 가능하다.

ML-Agents의 Agent 클래스는 표 2에서와 같이 학습을 위해 필요한 기본 메서드들을 제공하고 있다. Agent클래스를 통해서 에이전트와 환경의 기본 초기화, 관측을 통한 학습 정보 수집과 그에 따른 행동 수행, 그리고 보상 부여까지 가능하다.

표 2. Agent 클래스의 핵심 메서드
Table 2. Core methods of Agent class

Method	Description
Initialize()	Called once at the start to set up agent-specific variables.
CollectObservations(VectorSensor sensor)	Used to gather environment data (observations) for the agent's decision-making.
OnActionRecieved(ActionBuffers actionBuffers)	Executes agent actions received from the neural network.
OnEpisodeBegin	Called at the start of each new episode to reset agent states or environment.
Heuristics(in ActionBuffers actionsOut)	Defines manual (non-ML) controls for testing the agent without training.

2-3 알고리즘 선택

ML-Agents에서는 다양한 강화학습 알고리즘들을 지원한다. PPO, SAC 등의 Deep Reinforcement Learning 알고리즘들을 지원하고, Behavior Cloning과 같은 모방학습 알고리즘도 사용이 가능하다. 패키지에서는 Imitation Learning, POCA, PPO, SAC의 yaml 파일을 기본으로 지원한다. 강화학습에서 널리 사용되는 알고리즘들로, 각각 고유한 강점과 특성을 지닌다.

SAC는 Actor-Critic 기반의 Off-Policy 알고리즘으로, 정책과 밸류 함수 두 가지를 학습하는 강화학습 알고리즘이다[9]. Maximum Entropy Framework를 사용하여, 에이전트가 보상뿐만 아니라 정책의 엔트로피를 최대화하도록 학습한다. 이는 탐험(Exploration)과 활용(Exploitation)을 더 균형있게 수행하도록 돕는다. Off-Policy 특성으로 인해 샘플 효율성이 높다는 장점을 지닌다. 반면, 계산 비용이 높고 구현이 비교적 복잡하며, 학습 초기에는 정책이 너무 무작위적이어서 수렴 속도가 느릴 수 있다는 단점을 가지고 있다.

POCA는 기존의 Off-Policy Actor Critic 기법들을 개선한 형태이다[10]. POCA는 특히 다중 에이전트 시스템에서 강점을 보이는 알고리즘이다. Pessimistic Critic Update를 통해 가치 함수의 업데이트 시 기대 보상을 보수적으로 추정한다. 이는 잘못된 보상 예측으로 인해 정책이 불안정하게 업데이트되는 것을 방지하며, 높은 학습 안정성을 보장한다. SAC와 마찬가지로 Actor-Critic 기반의 Off-Policy 알고리즘으로, 데이터 재사용이 가능하고 샘플 효율성이 높다.

PPO는 Actor-Critic 기반의 On-Policy 알고리즘이다[11]. 정책을 직접 업데이트 하는 Policy Gradient 방법의 안정성과 효율성을 높인 알고리즘이다. 기존 정책과 새로 학습된 정책 간의 변화폭을 제한하는 Clipping 기법을 사용해 안정성을 확보한다. 비록 샘플 효율성이 앞의 두 알고리즘보다 낮고 복잡한 고차원 환경에서는 SAC에 비해 성능이 낮을 수 있지만, 구현이 비교적 간단하고, 안정적으로 학습이 가능하며 성능이 일관되고 견고하다는 강점을 지니고 있어 ML-Agents에서 가장 널리 사용되는 알고리즘이다.

표 3. 알고리즘 비교
Table 2. Algorithms comparison

Feature	PPO	SAC	POCA
Learning method	On-policy	Off-policy	Off-policy
Learning stability	Stable with clipping	Stable with entropy regularization	Stable through pessimistic critic updates
Sample efficiency	Low	High	High
Stable environments	Simple and stable environments	Continuous, high-dimensional action	Multi-agent, high-uncertainty environments
Multi-agent support	Limited	Limited	Strong

표 3에서 알고리즘들을 간단하게 비교하여 살펴보았고, 본 연구에서는 PPO 알고리즘을 주력으로 사용한다. ML-Agents에서 PPO 알고리즘은 가장 기본적으로 사용되는 정책 기반 알고리즘이다. 정책 기반 알고리즘은 에이전트가 환경과 상호작용하면서 정책을 업데이트할 때 큰 변화없이 안정적으로 최적화를 수행한다. 구현이 다른 강화학습 알고리즘에 비해 간단하고 효과적인 성능을 보이며, 여러 환경에서 병렬 학습이 가능하기 때문에 학습 진행에 있어서 효율적이다.

2-4 파괴 시뮬레이션의 게임 적용 사례 분석

물체의 파괴 시뮬레이션은 많은 현대 게임에서 중요한 요소로 자리잡고 있다[12]. 특히, 물리 기반의 파괴 시스템은 게임의 현실감을 높이고 몰입감을 증대시킨다. 예를 들어, FPS 게임에서 벽이나 건물이 무너지는 장면은 전투의 긴장감을 극대화하며, 오픈 월드 게임에서는 플레이어의 행동에 따라 환경이 동적으로 변화하는 경험을 제공한다. 이러한 파괴 시뮬레이션은 단순한 비주얼 효과를 넘어, 게임플레이의

전략적 요소로도 작용하며, 게임의 상호작용성을 한층 강화하고 있다.

1) 톰 클랜시의 레인보우 식스 시즈

"톰 클랜시의 레인보우식스 시즈" 는 전략적 요소를 강조한 경쟁 FPS게임으로, 파괴 시뮬레이션을 중요한 게임플레이 메커니즘으로 활용한다. 이 게임에서 플레이어는 공격팀과 방어팀으로 나뉘어 전투를 벌이며, 각 팀은 다양한 전술을 통해 목표를 달성해야 한다. 특히, 레인보우식스 시즈는 벽, 바리케이드, 바닥, 천당 등 다양한 환경 요소가 파괴 가능하게 설계되어 있으며, 이러한 파괴가 곧 게임의 전략적 요소로 작용한다. 공격팀은 벽을 폭파해 새로운 진입로를 만들거나, 방어팀의 시야를 차단해 기습을 시도할 수 있고, 방어팀은 파괴된 벽을 활용해 공격팀의 움직임을 추적하거나 새로운 방어 포지션을 확보할 수 있다. 이처럼 파괴 가능한 환경은 플레이어가 상황에 맞춰 유연한 전략을 세울 수 있게 하며, 단순한 사격 실력을 넘어 창의적인 플레이가 요구된다. 이러한 요소는 레인보우식스 시즈의 핵심 매력 중 하나로, 매 경기마다 다른 전개를 만들어내는 중요한 변수가 된다.

그림 2. 레인보우 식스 시즈의 예시[13]
Fig. 2. Examples of Rainbow Six Siege[13]

2) 배틀필드 시리즈

"배틀필드 시리즈"는 사실적인 전쟁 표현을 강조한 경쟁 FPS게임으로, 대규모 인원이 참여하는 전투와 분대 시스템을 통해 마치 실제 전쟁터에 있는 듯한 경험을 제공하는 것이 특징이다. 이 게임에서 파괴 시뮬레이션은 몰입감과 현장감을 극대화하는 중요한 요소로 작용한다. 건물, 다리, 벽 등 다양한 환경 요소가 실제와 같이 파괴되며, 플레이어는 전투 중 주변 지형이 동적으로 변하는 상황을 직접 체험하게 된다. 이러한 파괴 요소는 단순한 시각적 효과를 넘어서, 전투의 전략적 상황에도 영향을 미친다. 예를 들어, 건물이 무너져 새로운 길이 생기거나, 엄폐물이 사라져 전투의 양상이 급격히 변화할 수 있다. 또한 분대 시스템을 통해 협력하며 목표를 달성하는 과정에서, 파괴된 환경을 적절히 활용하는 것이 중요한 전술적 요소가 된다. 배틀필드 시리즈의 파괴 시뮬레이션은 실제 전쟁 속 병사가 된 것 같은 긴장감을 제공하며, 전장의 현실감을 한층 더 높여준다.

그림 3. 배틀필드3[14]
Fig. 3. Battlefield 3[14]

두 사례인 "톰 클랜시의 레인보우식스 시즈"와 "배틀필드 시리즈"는 파괴 시뮬레이션이 게임에서 다양한 방식으로 활용될 수 있음을 보여준다. 레인보우식스 시즈에서는 파괴 시뮬레이션이 전략적 요소로 작용하여, 플레이어가 벽이나 바리케이드를 파괴하고 새로운 전술적 진입로를 만드는 등 전투의 판도를 바꾸는 핵심 역할을 한다. 이처럼 파괴는 단순한 비주얼 효과가 아니라, 창의적이고 유동적인 플레이를 요구하는 중요한 메커니즘이다. 반면, 배틀필드 시리즈에서는 파괴 시뮬레이션이 몰입감과 현장감을 극대화하는 데 사용된다.

이 두 사례를 통해 알 수 있듯이, 파괴 시뮬레이션은 단순한 그래픽적 요소를 넘어 게임의 전략성, 몰입감, 그리고 상호작용성을 강화하는 중요한 도구로서 활용될 수 있다. 각각의 게임이 이를 어떻게 활용하느냐에 따라 플레이어 경험은 크게 달라질 수 있으며, 이는 파괴 시뮬레이션이 현대 게임 디자인에서 점점 더 중요한 역할을 하고 있음을 시사한다.

2-5 기존 연구 사례 분석

물체 파괴 시뮬레이션은 현실 세계에서의 물리적 현상을 가상 환경에서 재현하기 위해 중요한 연구 분야로 자리 잡았다. 초기 연구들은 주로 물체 파괴의 기초적인 물리적 원리를 시뮬레이션하기 위해 힘과 응력의 분포를 계산하는 데에 중점을 두었다. 이러한 시뮬레이션은 유한 요소 해석(Finite Element Analysis)을 활용하여 구조적 손상을 예측하거나 물체의 변형 과정을 재현하는 방식으로 발전했다. O'Brien과 Hodgins[15]는 유한 요소 기법을 기반으로 한 물체 파괴 시뮬레이션 시스템을 개발하여, 파괴 과정이 물리 법칙에 기초한 사실적인 결과를 보여줄 수 있음을 입증했다.

이후 연구들은 계산 비용을 줄이고 시뮬레이션의 효율성을 높이는 방향으로 발전해왔다. Müller 등은 강체의 물리적 파괴를 표현하기 위해서 시각적 메시를 볼륨 기반 근사 블론 분해(Volumetric Approximate Convex Decompositions)로 표현하고, 충격 지점에 따라 사용자 정의 파괴 패턴을 적용했다. 이 방법은 객체의 여러 위치에 부분적 파괴를 만들어낼 수 있다[16].

특히 게임 및 영화 산업에서는 디지털 콘텐츠 제작에 최적화된 방법론이 연구되고 있다. Unity와 Unreal Engine과 같은 상용 게임 엔진은 쉐이더와 스크립트를 활용한 간소화된

파괴 시뮬레이션을 지원하며, 특히 Unity에서는 Voronoi 분할 알고리즘을 활용하여 물체를 다각형 조각으로 분해하고 이를 물리 엔진과 연동하여 파괴 시뮬레이션을 구현할 수 있다.

Voronoi 알고리즘은 공간을 여러 개의 다각형 셀로 분할하는 기법으로, 주어진 점들에 가장 가까운 영역을 계산하여 셀을 정의한다[17]. 파괴 시뮬레이션에서는 Voronoi 알고리즘을 활용하여 물체를 다각형 조각으로 분할하고, 각 조각에 물리적 속성을 부여하여 파괴 과정을 표현한다. 물리 기반의 사실적 파괴 시뮬레이션은 게임 환경에서 빠르게 동작하기 어려운 점[18]과 효율적인 계산 덕분에 많은 게임들이 Voronoi 알고리즘에 기반한 파괴 시뮬레이션을 구현해 사용한다. 반면, Voronoi 알고리즘은 생성점 간의 거리 기반으로 분할을 수행하기 때문에, 균질한 분할로 인한 비현실성과 부자연스러운 결과를 초래한다는 단점이 존재한다.

Voronoi 알고리즘 기반의 파괴 시뮬레이션은 계산 효율성과 시각적 자연스러움을 제공하는 강력한 도구이다. 하지만 복잡한 물리적 특성을 반영하거나 고유한 파괴 패턴을 요구하는 시뮬레이션에서는 한계를 보일 수 있다. 이를 보완하기 위해 본 연구에서는 머신러닝과 결합하여 더욱 효과적인 파괴 시뮬레이션을 구현한다.

그림 4. 20개의 점과 Voronoi cells[19]
Fig. 4. 20 points and their Voronoi cells[19]

Ⅲ. 학습환경 구성 및 학습 진행

3-1 시뮬레이션 환경 구성

표 4. 소프트웨어 버전 설명
Table 4. Software versions explanation

Software	Version
Unity	2022.3.28f1
ML-Agent Package	Release 21
Python	3.9.13
Pytorch	2.4.0+cpu
Tensorflow	2.17.1

유니티 엔진과 ml-agents를 사용하여 시뮬레이션 환경을 구축하였다. 외부 라이브러리 충돌에 대비하기 위해서 유니티 파일 내부에 python venv를 통해 가상환경을 구축하고, 강화학습과 관련된 Python API들을 설치하였다. 환경 구축에 사용된 소프트웨어 버전들은 표 3에서 확인할 수 있다.

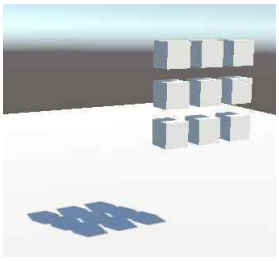

그림 5. 학습을 위한 환경 구성
Fig. 5. Configurating environment for learning

학습을 위한 유니티 Scene을 그림 5와 같이 구성하였다. 9개의 Agent가 존재하며, 이들의 중심 좌표에 충격이 가해지도록 설정하였다. 충격이 가해진 좌표를 중심으로 사전에 정의해둔 파라미터만큼의 Vertex가 새로 생성된다. 이 Vertex들과 충격 지점과의 거리를 관측 파라미터로 설정하고, 이 거리와 개발자가 별도로 설정한 수치와의 차이를 계산해서 Agent들이 Reward를 획득한다. 여기서 생성된 Vertex들의 정보를 토대로 새로운 Triangle과 Plane들을 생성하여 작은 메쉬 파편들을 생성하게 된다.

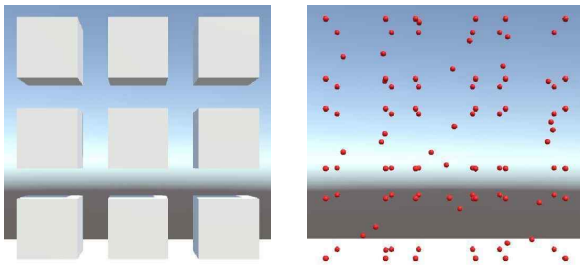

그림 6. Vertex 생성 학습 과정
Fig. 6. Vertex generation learning Process

3-2 스크립트 작성 및 파라미터 설정

파괴가 가능한 Destructable Object들에게 DestructibleAgent.cs 라는 스크립트를 작성해 컴포넌트로 붙여주었다. DestrucableAgent클래스는 ml-agents의 Agent클래스를 상속받아 만들어지는 클래스로, 통해서 생성된 버텍스와 충격점과의 거리를 관측한다. 그리고 그 거리에 따라 다른 보상을 부여한다. 거리와 보상에 따른 기준은 표 3을 따른다.

표 5. 에이전트가 받을 보상
Table 5. Amount of rewards of agent

Distance from target Vector (Normalized)	Rewards (Normalized)
7~6	−2
6~5	−1
5~4	1
4~2	2
2~1	3
1~0	−1

가장 중요하게 여기는 관측 정보가 바로 충격 지점과의 거리이고, 에이전트가 받을 보상을 결정하는 가장 중요한 요인이 된다. 이 과정을 통해서 에이전트는 충격점에 따라 적절한 위치에 정점들을 만들고, 정점 배열 정보를 업데이트한다. 업데이트가 완료되면 해당 에피소드는 종료되고, 새로운 에피소드를 시작하여 다시 학습을 진행한다.

3-3 파괴 시뮬레이션 적용

메쉬 파괴 시뮬레이션은 3D 오브젝트를 여러 조각으로 분할하고, 각 조각에 물리적 힘을 가하여 흩어지도록 하는 시뮬레이션을 수행한다. 본 연구에서 사용한 알고리즘은 Unity 기반으로 설계되었으며, 메쉬의 삼각형 데이터를 기반으로 평면을 생성하고 이를 기준으로 메쉬를 나눈다. 각 파편에는 물리적 속성이 부여되며, 지정된 폭발력에 의해 흩어지는 동작을 구현한다.

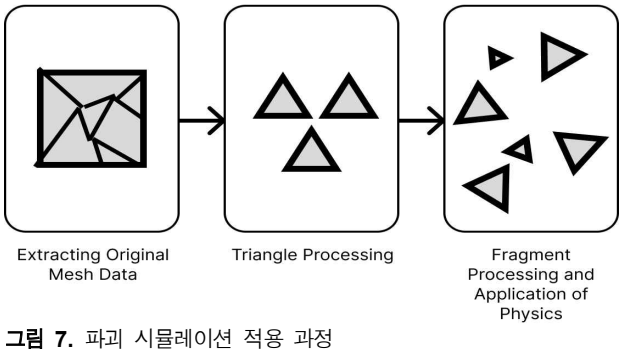

그림 7. 파괴 시뮬레이션 적용 과정
Fig. 7. Process of mesh destroy

그림 7에서와 같이 이 알고리즘은 크게 세 단계로 이루어진다. 첫 번째 단계는 원본 메쉬 데이터를 추출하는 과정이다. 이 과정에서 MeshFilter 컴포넌트를 통해 원본 메쉬의 Triangle, Vertex, Normal, UV 좌표 등의 정보를 획득한다. 이후, Plane 클래스를 이용해 메쉬를 나눌 랜덤 평면을 생성한다. 평면은 유니티의 Random API를 통해 임의의 방향을 갖도록 생성되며, 이 평면이 원본 메쉬의 경계를 따라 랜덤한 위치에 배치된다. 이 때, 평면을 기준으로 메쉬의 Triangle들이 두 개의 그룹으로 나뉘게 된다.

두 번째 단계에서는 평면과 교차하는 Triangle을 처리한다. 각 Triangle의 세 점 중 두 점이 평면의 다른 쪽에 위치할 때, 평면과 교차하는 지점을 계산하여 새로운 꼭짓점을 생성한다. 이 새로운 꼭짓점들을 기준으로 메쉬의 새로운 Triangle이 구성되며, 두 개의 파편 메쉬가 만들어진다. 메쉬가 평면과 교차하지 않는 경우, 해당 Triangle은 그대로 하나의 파편으로 할당된다. 이 과정에서 평면과 Triangle 간의 교차점을 계산하며, Triangle을 평면의 왼쪽 또는 오른쪽에 할당하여 파편을 나눈다.

세 번째 단계는 각 파편을 새로운 Unity 오브젝트로 변환하는 과정이다. 분할된 파편은 각기 새로운 오브젝트로 생성되며, 각각의 파편에는 Rigidbody, MeshRenderer와 같은 Unity 컴포넌트가 추가된다. 특히, Rigidbody 컴포넌트는 파편에게 물리적 속성을 부여하며, 이를 통해 파편은 중력의 영향을 받을 수 있으며 폭발력에 의해 흩어질 수 있다.

3-4 구현 결과

강화학습을 활용하여 물체의 파괴를 시뮬레이션한 결과들을 그림 8, 그림 9, 그림 10을 통해 확인할 수 있다. 그림 8과 그림 9에서는 게임 환경에서 일반적으로 관찰되는 물체 파괴의 형태를 학습한 결과를 확인할 수 있다. 기본적인 충격과 그로 인한 파괴 과정을 시뮬레이션 함으로써 실제 물리적 환경에서 나타날 수 있는 파괴 패턴을 재현했다.

그림 10에서는 파괴될 때 파편들이 일정한 패턴을 유지하도록 학습시킨 결과로써, 각 파편들이 동일한 모양과 동일한 크기를 가능한 유지하도록 만들어냈다. 또한, 그림 11에서는 특정 충격지점을 중심으로 메쉬가 방사형으로 퍼져나가는 과정을 확인할 수 있다. 이를 통해서 강화학습 모델이 충격의 중심과 주변 영역에서의 파괴 분포를 효과적으로 학습했음을 알 수 있다.

그림 8, 그림 9과 같은 일반적인 파괴 시뮬레이션과 그림 10와 그림 11과 같이 특정 효과들을 넣은 파괴 시뮬레이션을 구현했다. 이를 통해 기존에 무작위한 파편들을 만들어 적용되는 일반적인 파괴 시뮬레이션들과는 다르게, 특정 충격점을 적용하여 사방으로 흩어지듯이 폭발하는 파괴 시뮬레이션을 만들어내거나, 파편들의 모양을 일정하게 표현하도록 연출할 수 있음을 확인할 수 있다.

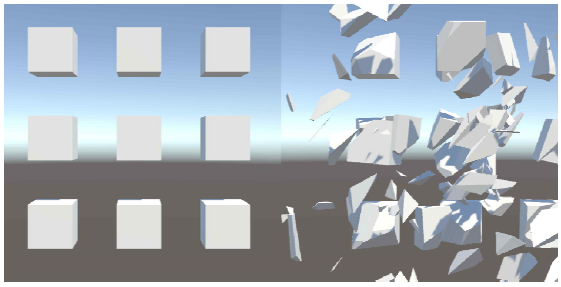

그림 8. 일반 파괴 학습 결과1
Fig. 8. Learning result 1 (normal)

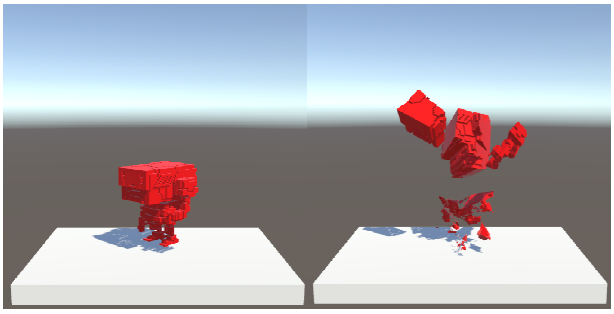

그림 9. 일반 파괴 학습 결과2
Fig. 9. Learning result 2 (normal)

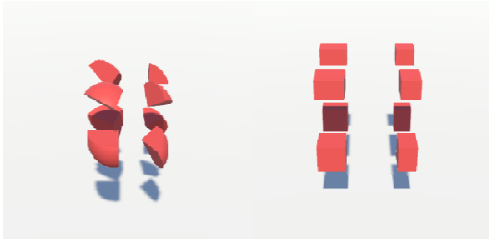

그림 10. 일정한 형태와 크기의 파편
Fig. 10. Fragments of uniform shpae and size

그림 11. 방사형 파괴 학습 결과
Fig. 11. Learning result (radial)

3-5 기존 기법과의 비교

본 연구에서는 Voronoi 알고리즘 기반의 파괴 시뮬레이션이 가진 한계점인 비현실적이고 균질한 분할 구조, 그리고 고유한 파괴 패턴을 구현하기 어려운 문제를 보완하기 위한 방법을 제안하였다. 기존의 Voronoi 알고리즘은 생성점 간의 거리 기반으로 단순하게 공간을 분할하기 때문에, 물리적 특성, 충격 방향과 같은 세부적인 요소를 충분히 반영하지 못하였다. 이를 해결하기 위해 본 연구에서는 강화학습 기법을 활용하여 충격 지점과 특정한 파괴 패턴을 만들어낼 수 있도록 구현하였다. 이 기법은 충격의 세기와 방향에 따라 메쉬의 정점을 동적으로 배치하며, 이를 통해 조각의 크기와 형태가 물리적 상황에 따라 유연하게 변화하도록 하였다. 또한, 강화학습 기반의 알고리즘을 사용하여 파괴 과정에서 발생하는 균열 패턴과 파편 분포를 학습함으로써, 자연스럽고 고유한 파괴 결과를 생성할 수 있도록 설계하였다.

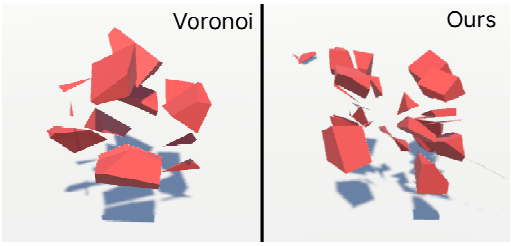

그림 12. Voronoi 기반과 본 연구 방식의 비교
Fig. 12. Comparison between Voronoi-based method and the proposed method

충격 지점이 오브젝트의 중점인 상황에서, Voronoi 알고리즘 기반으로 구현된 파괴 시뮬레이션과 본 연구의 방법으로 구현된 파괴 시뮬레이션의 차이를 그림 12를 통해 확인할 수 있다. 해당 그림은 충격 조건에 따라 생성되는 고유한 파괴 패턴 생성과 자연스러운 파편 분포를 보여주며, Voronoi 기반 파괴 시뮬레이션의 약점을 보완할 수 있음을 확인할 수 있었다. 또한 그림 13을 통해서 실제 게임에까지 적용될 수 있음을 확인할 수 있다.

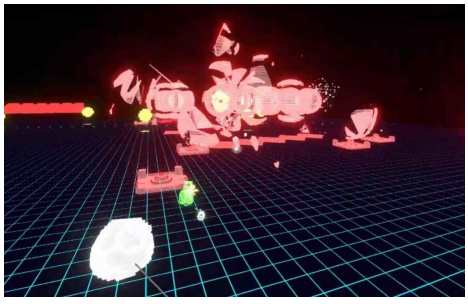

그림 13. 게임 적용 예시
Fig. 13. Game application example

Ⅳ. 결 론

본 연구에서는 강화학습을 활용하여 물체의 충격점에 따른 메쉬 정점들을 생성하고, 이를 기반으로 파괴 알고리즘을 적용해 물체 파괴 시뮬레이션을 성공적으로 구현했다. 메쉬 정점 생성 과정에서 강화학습을 활용한 것은, 기존의 파괴 시뮬레이션보다 다양한 파괴 양상을 구현할 수 있게 해준다는 점에서 큰 의미가 있다. 이는 단순한 시각적 효과를 넘어서 게임적 연출 표현을 풍부하게 만들어줄 수 있다.

본 연구에서 본 것과 같이 파괴 시뮬레이션은 게임 속 다양한 상황에서 적용이 가능하다. “톰 클랜시의 레인보우식스 시즈” 와 “배틀필드 시리즈”에서 사례로 확인한 바와 같이, 파괴 시뮬레이션은 단순히 그래픽적 요소로 사용되는 것을 넘어 게임플레이의 중요한 전략적 요소로 작용할 수 있다. 레인보우식스 시즈에서는 플레이어가 벽을 파괴하여 새로운 전술

적 진입로를 만들거나, 적의 시야를 차단하는 전략적 요소로 활용되고 있으며, 배틀필드 시리즈에서는 현실감을 극대화하고 몰입도를 높이는 데 사용되고 있다. 이러한 사례들은 파괴 시뮬레이션이 게임 내에서 다양한 방식으로 사용될 수 있음을 보여주고 있다.

본 연구에서 제안한 강화학습 기반의 파괴 시뮬레이션은 이러한 게임의 상호작용성 및 현실성을 발전시키는 데 기여할 수 있을 것이라고 생각한다. 특히, 강화학습을 통해 파괴 과정을 보다 동적인 방식으로 구현할 수 있게 되면서, 기존의 고정된 파괴 시뮬레이션보다 훨씬 더 유연하고 예측할 수 없는 상황을 연출할 수 있다는 장점이 있다. 이는 게임뿐만 아니라, 시뮬레이션을 필요로 하는 다양한 분야에서 응용될 수 있을 것이다. 예를 들어, 가상현실(VR) 환경에서도 파괴 시뮬레이션을 더욱 현실감 있게 구현할 수 있으며, 영화나 애니메이션의 파괴 연출에서도 활용될 수 있다.

결론적으로, 본 연구에서는 강화학습을 통한 메쉬 정점 생성 및 물체 파괴 알고리즘을 구현했으며, 이를 통해 게임뿐만 아니라 여러 시뮬레이션 환경에서의 가능성을 확인할 수 있었다. 앞으로 파괴 시뮬레이션은 더욱 다양한 방식으로 발전하고, 게임 디자인의 중요한 요소로서 계속 활용될 것이다. 또한 이번 연구가 그러한 발전에 일조하는 연구가 되기를 기대한다. 강화학습을 통한 물체 파괴의 유연한 연출은 향후 더 많은 연구와 응용을 통해 더욱 발전할 것이며, 게임뿐만 아니라 다양한 디지털 환경에서 적용할 수 있을 것이다.

감사의 글

본 연구는 과학기술정보통신부 및 정보통신기획평가원의 정보통신방송혁신인재양성(메타버스융합대학원)사업 연구결과로 수행되었습니다(IITP-2024-RS-2024-00425383).

참고문헌

[1] M. Müller and N. Chentanez, “Solid Simulation with Oriented Particles,” *ACM Transactions on Graphics*, Vol. 30, No. 4, 92, July 2011. https://doi.org/10.1145/2010324.1964987

[2] M. Müller, R. Keiser, A. Nealen, M. Pauly, M. Gross, and M. Alexa, “Point Based Animation of Elastic, Plastic and Melting Objects,” in *Proceedings of the 2004 ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA ’04)*, Grenoble, France, pp. 141-151, August 2004. https://doi.org/10.1145/1028523.1028542

[3] K. Souchleris, G. K. Sidiropoulos, and G. A. Papakostas, “Reinforcement Learning in Game Industry—Review, Prospects and Challenges,” *Applied Sciences*, Vol. 13, No. 4, 2443, February 2023. https://doi.org/10.3390/app13042443

[4] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd ed. Cambridge, MA: The MIT Press, 2018.

[5] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, “Deterministic Policy Gradient Algorithms,” in *Proceedings of the 31st International Conference on Machine Learning (ICML 2014)*, Beijing, China, pp. 387-395, June 2014.

[6] L. P. Kaelbling, M. L. Littman, and A. W. Moore, “Reinforcement Learning: A Survey,” *Journal of Artificial Intelligence Research,* Vol. 4, pp. 237-285, May 1996. https://doi.org/10.1613/jair.301

[7] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, ... and D. Hassabis, “Human-Level Control through Deep Reinforcement Learning,” *Nature*, Vol. 518, No. 7540, pp. 529-533, February 2015. https://doi.org/10.1038/nature14236

[8] Unity. Unity ML-Agents Toolkit Download Page in Github [Internet]. Available: https://github.com/Unity-Technologies/ml-agents.

[9] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, “Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor,” arXiv:1801.01290v2, August 2018. https://doi.org/10.48550/arXiv.1801.01290

[10] J. Inman, T. Khandait, G. Pedrielli, and L. Sankar, “Parameter Optimization with Conscious Allocation (POCA),” in *Proceedings of 2023 Winter Simulation Conference (WSC)*, San Antonio: TX, pp. 3436-3447, December 2023. https://doi.org/10.1109/WSC60868.2023.10407962

[11] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, “Proximal Policy Optimization Algorithms,” arXiv:1707.06347, 2017. https://doi.org/10.48550/arXiv.1707.06347

[12] Z. Bao, J. Hong, J. Teran, and R. Fedkiw, “Fracturing Rigid Materials,” *IEEE Transactions on Visualization and Computer Graphics*, Vol. 13, No. 2, pp. 370-378, March-April 2007. https://doi.org/10.1109/TVCG.2007.39

[13] Esports Tales. Reinforcement and Barricade Placements - Strategy Guide [Internet]. Available: https://www.esportstales.com/rainbow-six-siege/reinforcement-and-barricade-placements-strategy-guide.

[14] Steam. Battlefield 3™ on Steam [Internet]. Available: https://store.steampowered.com/app/1238820/Battlefield_3/.

[15] J. F. O’Brien and J. K. Hodgins, “Graphical Modeling and Animation of Brittle Fracture,” in *Proceedings of the 26th*

Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '99), Los Angeles: CA, pp. 137-146, August 1999. https://doi.org/10.1145/311535.311550

[16] W. Pokojski and P. Pokojska, "Voronoi Diagrams – Inventor, Method, Applications," *Polish Cartographical Review*, Vol. 50, No. 3, pp. 141-150, September 2018. https://doi.org/10.2478/pcr-2018-0009

[17] M. Müller, N. Chentanez, and T.-Y. Kim, "Real Time Dynamic Fracture with Volumetric Approximate Convex Decompositions," *ACM Transactions on Graphics*, Vol. 32, No. 4, 115, July 2013. https://doi.org/10.1145/2461912.2461934

[18] S. Sellán, J. Luong, L. M. Da Silva, A. Ramakrishnan, Y. Yang, and A. Jacobson, "Breaking Good: Fracture Modes for Realtime Destruction," *ACM Transactions on Graphics*, Vol. 42, No. 1, 10, February 2023. https://doi.org/10.1145/3549540

[19] Wikipedia. Voronoi Diagram [Internet]. Available: https://en.wikipedia.org/wiki/Voronoi_diagram.

문철호(Cheol-Ho Mun)

2024년 : 경희대학교 소프트웨어융합학과 (학사)

2018년 ~ 현 재: 경희대학교 소프트웨어융합학과 학사과정

※관심분야 : 게임 콘텐츠(Game Contents), 머신러닝(Machine Learning)

우탁(Tack Woo)

2002년 : University of Dundee (UK), Electronic Imaging. BA (Honours)

2004년 : University of Dundee (UK), Electronic Imaging. MSc (이학석사)

2010년 : University of Dundee (UK), Electronic Imaging. (게임학), PhD (이학박사)

2004년 ~ 2007년: University of Dundee, Lecturer

2007년 ~ 2010년: KAIST 엔터테인먼트 공학연구소, 연구원 (기능성 게임랩)

2010년 ~ 2012년: KAIST 문화기술대학원, Digital Art & Entertainment Track 초빙교수 (게임)

2012년 ~ 2013년: 서울대학교 차세대융합기술연구원, 게임융합미디어연구센터 센터장

2013년 ~ 2024년: 경희대학교 디지털콘텐츠학과 교수

2024년 ~ 현 재: 경희대학교 일반대학원 메타버스학과 교수

※관심분야 : 기능성 게임, 게임화, 게임문화, VR/AR 콘텐츠