KOSDI
ISSN : 1976-2208 (Print)
ISSN : 2671-5287 (On-line)
www.sodi.or.kr

Journal of the Society of Disaster Information
Vol. 16, No. 4, pp. 813-823, December 2020
https://doi.org/10.15683/kosdi.2020.12.31.813

Original Article

디지털 트윈을 적용한 고감도 충돌 시뮬레이션 개발을 위한 연구

A Study on the Development of High Sensitivity Collision Simulation with Digital Twin

기재석[1] · 황교찬[2] · 최주호[3]*
Jae-Sug Ki[1], Kyo-Chan Hwang[2], Ju-Ho Choi[3]*

[1]Professor, Department of Sports ICT Convergence Sangmyung University, Seoul, Republic of Korea
[2]Master's and Ph.D., Department of Sports ICT Convergence Sangmyung University, Seoul, Republic of Korea
[3]Researcher, Future Innovation Convergence Technology Research Institute Sangmyung University, Seoul, Republic of Korea

*Corresponding author: Ju-Ho Choi, kchoi612@gmail.com

ABSTRACT

Purpose: In order to maximize the stability and productivity of the work through simulation prior to high-risk facilities and high-cost work such as dismantling the facilities inside the reactor, we intend to use digital twin technology that can be closely controlled by simulating the specifications of the actual control equipment. Motion control errors, which can be caused by the time gap between precision control equipment and simulation in applying digital twin technology, can cause hazards such as collisions between hazardous facilities and control equipment. In order to eliminate and control these situations, prior research is needed. **Method:** Unity 3D is currently the most popular engine used to develop simulations. However, there are control errors that can be caused by time correction within Unity 3D engines. The error is expected in many environments and may vary depending on the development environment, such as system specifications. To demonstrate this, we develop crash simulations using Unity 3D engines, which conduct collision experiments under various conditions, organize and analyze the resulting results, and derive tolerances for precision control equipment based on them. **Result:** In experiments with collision experiment simulation, the time correction in 1/1000 seconds of an engine internal function call results in a unit-hour distance error in the movement control of the collision objects and the distance error is proportional to the velocity of the collision. **Conclusion:** Remote decomposition simulators using digital twin technology are considered to require limitations of the speed of movement according to the required precision of the precision control devices in the hardware and software environment and manual control. In addition, the size of modeling data such as system development environment, hardware specifications and simulations imitated control equipment and facilities must also be taken into account, available and acceptable errors of operational control equipment and the speed required of work.

Keywords: Simulation, Collision Check, Digital Twin, Collision Experiment, Virtual Environment

요 약

연구목적: 원자로 내부 시설물 해체 등의 고위험 시설이나 고비용 작업에 앞서 시뮬레이션을 통한 작업의 안정성과 생산성을 최대화하기 위해 실제 제어 장비의 제원을 시뮬레이션 상에 모사하고 이를 통해 정밀 제어될 수 있는 디지털 트윈 기술을 이용하고자 한다. 디지털 트윈 기술을 적용함에 정밀 제어 장비와 시뮬레이션의 시간적 격차로 인해 발생할 수 있는 동작 제어 오차는 위험 시설물과 제어 장비 간의 충돌 등과 같은 위험 요소들을 발생시킬 수 있다. 이러한 상황을 제거하고 통제하기 위해서는 사전 연구가 필요하다. **연구방법:** 현재 시뮬레이션을 개발함에 가장 대중적으로 사용되는 엔진으로는 Unity 3D

Received | 19 November, 2020
Revised | 9 December, 2020
Accepted | 17 December, 2020

가 있다. 하지만 Unity 3D 엔진 내부의 시간 보정으로 인해 발생할 수 있는 제어 오차가 존재한다. 그 오차는 여러 환경에 예상되고 그 오차는 시스템 사양 등의 개발 환경에 따라 다를 수 있다. 이를 입증하기 위해 Unity 3D 엔진을 이용하여 충돌 시뮬레이션 개발하고 이를 통해 다양한 조건의 충돌실험을 진행하고 그에 따른 결과를 정리하고 분석하여 이를 토대로 정밀 제어 장비의 허용 오차를 도출한다.
연구결과: 충돌실험 시뮬레이션을 통한 실험에서 엔진 내부 함수호출에 1/1000초 단위의 시간 보정으로 인해 충돌체의 이동제어에 단위 시간당 거리오차가 발생하고 거리오차는 충돌체의 이동속도와 비례한다. **결론:** 디지털 트윈을 이용한 원격해체 시뮬레이터는 하드웨어와 소프트웨어 환경과 수동 제어 시 정밀 제어 장치의 요구 정밀도에 따른 이동속도의 제한이 필요할 것으로 판단된다. 그리고 운용 제어 장비의 가용 및 허용 오차와 작업의 요구 속도를 시스템 개발 환경, 하드웨어 사양과 시뮬레이션에 모사된 제어 장비 및 시설물 등의 모델링 데이터의 크기도 반드시 고려한다.

핵심용어: 시뮬레이션, 충돌 체크, 디지털 트윈, 충돌실험, 가상환경

서 론

이론적 배경

디지털 트윈(Digital Twin)

디지털 트윈은 가상환경(VE : Virtual Environment)에 실제 물체와 동일한 가상의 물체를 소프트웨어를 이용해 가상화한 후 다양한 모의시험을 통해 검증해 보는 기술을 말한다. 이 개념은 미국 가전업체인 제너럴 일렉트릭(GE : General Electric)이 만들어 2000년대 들어 제조업을 시작으로 항공, 건설, 자동차, 국방 등의 다양한 산업 분야에서 폭넓게 활용되고 3차원 설계 프로그램을 사용하고 사물인터넷(IoT)을 통해 많은 정보를 수집할 수 있게 되면서 디지털 트윈의 정확도가 높아지고 진화하고 있다.

디지털 트윈 기술을 활용하면 가상세계에서 장비, 시스템 등의 상태를 모니터링하고 유지·보수 시점을 파악해 개선할 수 있다. 가동 중에 발생할 수 있는 다양한 상황을 예측해 안전을 검증하거나 돌발 사고를 예방해 사고 위험을 줄일 수도 있다. 또한 생산성 향상, 장비 최적화 등의 결과를 가져올 수 있고 시제품 제작에 들어가는 비용과 시간을 대폭 절감할 수 있다.

디지털 트윈으로 가상의 환경과 실제 환경의 일치도가 정밀 제어의 신뢰도 오차범위 이내라면 보다 정밀한 정확도를 요구하는 원전 시실 해체 현장에서도 큰 활약을 할 것이며 그로 인해 인간을 대체한 로봇이나 해체 장비의 활용으로 방사능 작업장에 노출되는 작업자의 피폭은 최소화될 것이다.

이와 같은 활용 범위와 제어환경의 일치도는 디지털 트윈의 핵심 기능이며 지속적인 개발과 고도화, 검증을 통해 신뢰 높은 기술의 적용을 보장받을 것이다.

유니티(Unity)

최근 다양한 게임을 실행하면 첫 화면에 유니티의 로고가 등장한다. 유니티는 모바일 게임뿐만 아니라 다양한 분야에서 널리 사용되고 있다. 유니티는 유니티 테크놀로지스(Unity Technolies, 이하 유니티) 사가 2004년에 캐주얼 게임과 온라인 게임 산업을 겨냥해 개발한 게임 엔진이고 당시 사용자의 대부분이 취미로 게임을 개발하는 사람이나 인디 개발자 정도였다. 이후 게임 업계에서 스타트업 기업으로 여겨졌던 유니티는 스마트폰 등의 모바일 장치의 등장으로 크게 성장할 수 있는 계기가 되었다. 2012년부터 유니티의 가입자 수는 100만명을 넘었고, 유니티로 개발된 여러 가지 게임들이 상업적으로 큰 성과를 이루며 업계에 입지를 굳혀갔다. 유니티는 게임 엔진 기술이자 통합개발환경(Integrated Development Environment,

IDE)이다. 이 기술을 처음 만든 이는 데이비드 헬가손(David Helgason), 니콜라스 프렌시스(Nicholas Fran-cis), 요하킴 안테(Joachim Ante)이다.

유니티를 개발 도구로 사용하는 가장 큰 이유는 25개 이상의 다양한 주요 플랫폼과 기술을 지원한다는 것이다. 모바일 장치, 웹, 콘솔 등의 다양한 형태의 환경에서도 직관적이고 간단하게 개발이 가능할 뿐만 아니라 잠재된 사용자의 능력을 이끌어낼 수 있다. 그리고 유니티에는 무료 버전과 유료 버전이 있다. 하지만 무료 버전이라고 기능 제한이 있는 것은 아니다. 이 정도의 기능과 성능을 가진 소프트웨어를 무료로 이용할 수 있다는 점은 큰 장점 중에 하나다. 이런 장점들로 게임 개발자뿐만 아니라 공학, 예술이나 교육 등의 다양한 분야에서 각각의 용도에 맞는 결과물을 개발하기 위해 유니티를 사용한다.

Unity 3D 엔진을 이용한 시뮬레이션 개발과 연구 사례

많은 양의 레이더 관측정보와 대용량 지형정보를 매쉬업하여 서비스하는 것은 데이터 처리에 있어 과부하가 발생하여 서비스 수준이 낮아지는 경향이 있다. 특히 위성영상, DEM 등을 활용한 3차원 지형정보는 그 자체만으로도 대용량 정보로 분류되고 있어서 신속한 서비스 구현을 위해 비교적 용량이 가벼운 2차원 정보를 활용하는 것이 일반적이기에 최근 모바일 게임 분야에서 활발히 활용 중인 Unity 3D 엔진을 사용한 3차원 지형기반 시각화 시스템 프로토타입을 구축하였다(Choi et al., 2015).

Compute shader를 통한 병렬 연산을 이용하면 효율적으로 구동할 수 있어 다양한 플랫폼을 지원하는 디지털 콘텐츠 제작 툴인 Unity와 다양한 플랫폼에서 구동되어지는 OpenGL에서의 실시간 물리 시뮬레이션에서의 성능을 측정 및 비교한다. 이를 통하여 추후 멀티 플랫폼을 지원하는 디지털 콘텐츠를 제작 함에 있어 더 나은 개발 도구를 선정할 수 있을 것으로 기대된다(Kim et al., 2017).

이 외에도 Unity 3D 엔진은 산업, 문화, 예술과 연구 등의 모든 분야에서 광범위하게 활용되고 많은 개발자와 사용자로 인해 엔진의 안정성 검증과 빠른 진화도 이루어지고 있다.

연구배경 및 목적

정부의 에너지전환 정책에 따라 40년간 전력을 공급해 온 고리1호기의 영구정지와 후속 조치인 월성1호기의 조기 폐쇄가 결정되며 현실적인 당면과제인 원전 해체의 안정성 확보를 위한 기술 확보가 중요한 현안이 되고 있다. 일반적으로 원전 시설 해체는 방사능으로 인한 작업에 피폭과 방사능 폐기물로부터의 안전성 확보가 핵심이다. 지난 2011년 일본 후쿠시마에서 발생한 원전 사고로 인해 전 세계적으로 원전 시설의 안전한 해체에 많은 관심을 가지게 되었으며 이에 따라 해체시장의 기술 선점을 위한 경쟁도 거세질 것으로 예상된다. 이미 해외 선진국들은 원전 등의 다양한 시설의 해체 경험과 기술의 안전성 및 경제성을 확보한 상태이며 이로 인해 국가적인 기술 종속이 발생할 것이다. 이러한 관점에서 원전 해체는 단순한 경제적 차원이 아닌 국가의 중장기 과제로 많은 지원이 필요하다. 이러한 상황에서 국내의 원전 해체는 실제 해체를 통해 해체기술의 국산화를 이룰 수 있는 적기이다.

국산 기술의 고도화를 위해 고위험 시설인 원전 시설을 원격으로 해체하고 가상환경에서 시뮬레이터를 통해 해체 장비를 제어하기 위해서는 장비의 정밀 제어 신뢰도를 기반으로 한 디지털 트윈 기술을 적용해야 한다. 이를 위해 정밀 제어를 가상의 환경에서 시뮬레이션 통해 다양한 형태와 조건에서 고감도 오브젝트 제어 및 충돌실험을 통해 정밀 제어의 신뢰도를 증명해야 한다.

이와 같은 일련의 과정을 통해 원격해체 가상 시뮬레이터를 통해 최소의 시간과 비용으로 원전 시설을 안전하고 효율적인 해체가 기술을 습득하고 해체 시나리오를 도출하고 검증하여 다양한 환경과 조건에서 발생할 수 있는 위험 요소나 제어의 결함 및 오작동 등을 사전에 검출하고 미연에 방지하여 고비용 시설물과 장비의 효율성은 극대화될 것이다.

이런 원격해체 가상 시뮬레이터를 개발하기 전에 정밀 장비 제어의 오차에 관한 신뢰도 연구가 앞서 진행되어야 하기에 Unity 3D 엔진으로 고감도 충돌실험 시뮬레이션을 개발하고 이를 이용하여 다양한 충돌 조건과 시스템 환경에서 Unity 3D 엔진 성능을 평가하고 분석하여 정밀 제어 장비의 요구 성능을 충족시킬 수 있는 기준치를 도출하는데 고감도 오브젝트 충돌실험의 목적이 있다.

물리 엔진의 정밀 제어 장비를 위한 최소 요구치

물리 엔진(Physics Engine)은 강체동역학, 연체동역학, 유동역학과 같은 물리현상을 시뮬레이션하는 소프트웨어이고 최근에는 실시간 게임이나 고성능 과학 시뮬레이션과 같은 상황을 충돌 판정과 물체의 동작을 제어하는 등의 물리 연산을 통해 구현한다.

Unity 3D 엔진도 다른 물리 엔진과 마찬가지로 하드웨어의 성능에 매우 의존적이고 제한적이다. 최신 하드웨어를 사용하더라도 시뮬레이션을 통해 실시간으로 모든 장비와 장치를 정밀 제어하고 고감도 충돌 체크 연산을 하는 것은 시스템에 물리적인 부하를 생성한다. 그리하여 로봇이나 정밀 제어 장비의 일부분에 대한 물리적인 충돌을 제어하고 테스트하여 동작 허용치를 정의해야 한다.

가상환경에서 해체 시뮬레이션의 요구 고감도 충돌 감지 정밀도는 10mm 이하여야 디지털 트윈을 적용하여 장비를 사용할 수 있다는 연구 논문도 있다(Chabal et al., 2011). 이처럼 디지털 트윈으로 제어되는 장비의 실시간 정밀 제어는 시간차로 발생하는 제어 오차를 검증하는 것이 디지털 트윈의 신뢰도에 핵심이다. 신뢰도를 검증하기 위한 충돌실험 시뮬레이션은 Unity 3D 엔진의 충돌 연산 방식을 연구해 충돌 체크 시 발생할 수 있는 상황과 조건을 예상하여 충돌 체크 실험을 진행한다.

본 론

Unity 3D 엔진을 이용한 고감도 충돌실험의 예측 결과

Unity 3D 엔진을 이용해 1/100초 단위의 충돌 체크 연산 과정의 함수 호출 시간 간격 정확도와 그에 따른 호출 함수에 의한 시간당 이동 거리를 측정함에 시간 보정에 따른 거리 오차가 발생할 것으로 예상된다. 그리고 Unity 3D 엔진으로 개발될 원전 시설 원격해체 시뮬레이터의 Manipulator와 End-Effector의 이동제어 시 제어되는 장비의 최대 이동속도와 시뮬레이터가 구동되는 시스템의 사양을 고려한 충돌실험의 단위 시간당 충돌 체크 범위의 효율성 및 안정성을 정의할 수 있다.

예상되는 충돌실험 결과를 토대로 요구되는 조건은 다음과 같이 예상된다. 디지털 트윈의 원격해체 시뮬레이터로부터 제어되는 Manipulator와 End-Effector의 충돌 감지 정밀도의 허용오차는 10mm 이하여야 하기에 그에 따른 장비의 제어 속도는 충돌 체크의 횟수에 따라 정의될 수 있다. 예를 들어 장비의 제어 이동속도가 100mm/sec 이하이고 충돌 체크는 초당 100회인 경우 충돌실험 시뮬레이션을 통해 이동속도의 허용오차를 예측하고 충돌실험 결과를 통해 원격해체 시뮬레이터의 Manipulator와 End-Effector의 정밀 제어 이동속도를 도출한다.

예측 결과를 도출하기 위한 시뮬레이션 개발

충돌실험 시뮬레이션 시스템 설명

본 연구에서 사용한 시뮬레이터는 Unity 3D 엔진을 이용하여 직접 연구 개발하여 구나 박스 등의 다양한 메쉬(Mesh: 3D 그래픽에서 그물망 형태로 만들어진 구조물)의 형태와 모델링 데이터를 선택하여 설정된 충돌 속도와 충돌 오브젝트의 방향 및 위치에 따른 충돌실험을 위한 프로그램이다. 기본적으로 Unity 3D 엔진에서 제공하는 Fixedupdate함수와 Collider Collision 체크를 이용하여 충돌 시 충돌체 간의 거리를 실시간으로 측정하고 이를 통해 거리 오차도 측정한다(Fan et al., 2016).

화면 설명에 앞서 Table 1과 같이 용어를 정리함에 본 논문에서 사용되는 충돌실험 시뮬레이션의 용어는 논문의 이해를 돕기 위해 자체적으로 정의하였으며 결과 분석 및 결론 도출에도 사용된다.

Table 1. Define terms for using「Collision Experiment Simulation」

시뮬레이션 용어	정의
Target Object	Replacing facilities to be dismantled, etc. as objects to be conflicted
Collider Object	Replacing precision control equipment, as a collision object moving toward a collision target
Rotation	Set the position direction value of the Target/Collider Object
Scale	Set the relative size of the Target/Collider Object (Default value is 1)
FPS(Frame Per Second)	Used as a function call or screen frame control unit in seconds to indicate the rate at which frames change

충돌실험 시뮬레이션: Target & Collider Object Setting 화면

Fig. 1을 참고하여 충돌체 설정 화면의 구조 및 기능을 설명하면 다음과 같다.

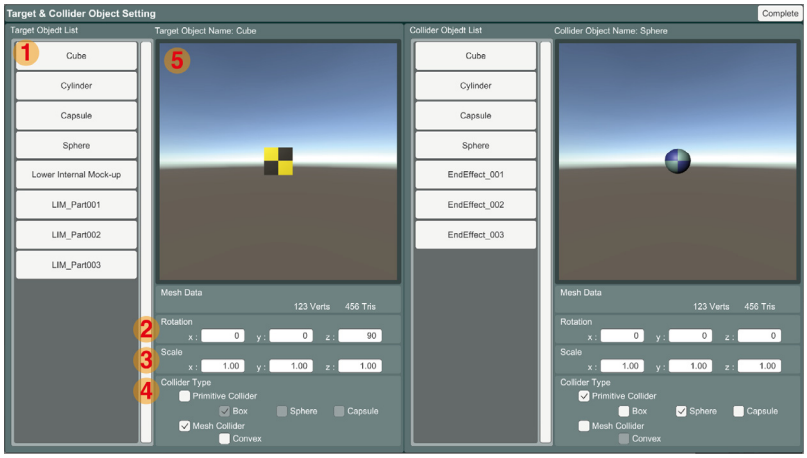


Fig. 1. Screen of「Target and collider object setting」

(1) Target/Collider Object List : 실험에 사용될 Object를 선택한다.

(2) Rotation : 오브젝트 간의 충돌이 발생하는 위치에 따른 선택된 Target Object Collider Object의 충돌 방향을 조정한다

(3) Scale : 선택된 Target Object나 Collider Object의 Collider Scale을 설정한다.

(4) Collider Type : 충돌 체크 정밀도를 선택한다.

(5) Preview : Target/Collider Object List에서 선택된 객체의 형태를 확인할 수 있다.

충돌실험 시뮬레이션: 충돌실험 화면

Fig. 2를 참고하여 충돌실험 화면구조와 기능을 설명하면 다음과 같다.

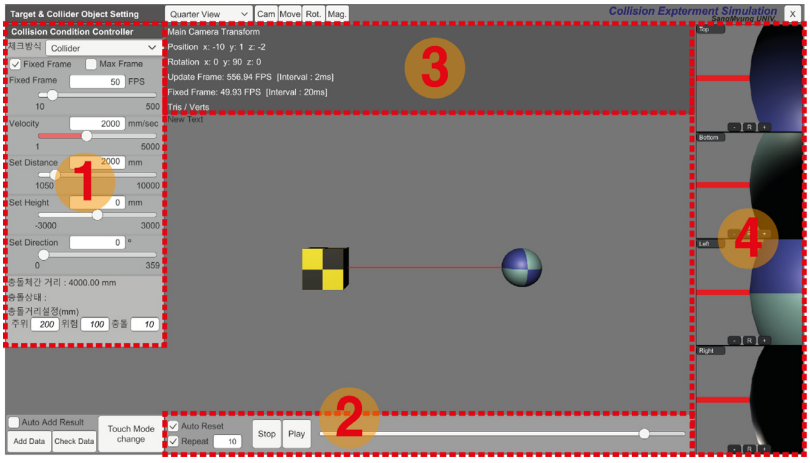


Fig. 2. Screen of「Collision experiment」

(1) Collision Condition Controller를 통해 충돌 조건을 설정한다.

(2) 충돌실험 진행 및 실험을 제어하는 역할을 한다.

(3) FPS 등의 상태 정보를 확인한다.

(4) Collider Object의 상하좌우에서 확대화면으로 충돌체 간의 충돌 여부를 확인한다.

Fig. 3은 충돌실험 시뮬레이션의 충돌실험 조건을 설정하는 부분으로 기능을 설명하면 다음과 같다.

(1) 방식 : Unity 3D 엔진에서 지원해주는 Mesh 오브젝트의 충돌을 확인하기 위해 지원하는 Collider를 이용한다.

(2) Fixed Frame : Unity 3D 엔진에 Fixedupdate 함수를 제어하여 단위 시간당 호출 횟수를 설정하여 충돌 체크의 횟수를 제어한다.

(3) Max Frame : 시스템의 사양에 따라 단위 시간당 최대한 많은 충돌 체크를 하여 발생할 수 있는 오차를 측정하고 충돌 체크 횟수를 실시간 확인이 가능하다.

(4) 이동/충돌 속도 : Collider Object의 이동속도를 1 ~ 5000 mm/sec 범위 내에서 설정 가능하며 이동 중에도 변경된 이동 속도가 실시간으로 적용된다.

(5) 초기 충돌 거리 : Target Object와 Collider Object 간의 충돌실험 전 거리를 설정하고 실험이 시작될 때 변경된 값이 적

용된다. 추가 수정이 없는 한 설정된 값으로 충돌실험의 Target Object와 Collider Object 간 거리가 자동 설정된다.

(6) 초기 충돌 높이 : Target Object의 충돌 위치를 Collider Object 높이로 설정한다.

(7) 초기 충돌 방향 : Target Object의 좌측이나 우측 부분을 충돌 지점으로 설정할 때 Collider Object의 이동 방향각을 설정한다.

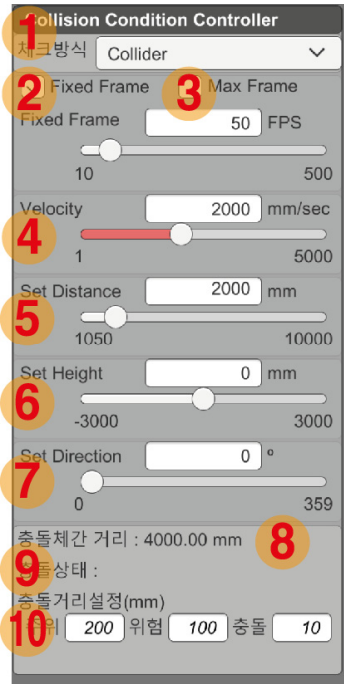


Fig. 3. Part of「Collision condition controller」

충돌실험 중의 충돌체 간의 상태 정보는 다음과 같다.

(8) 충돌체 간 거리 : 실험 시작 버튼을 누르면 Collider Object가 Target Object를 향해 설정된 이동속도와 방향으로 이동하며 두 Object 간의 간격을 실시간으로 표시한다. 충돌이 완료된 후에는 두 Object 간의 오차 거리를 나타낸다.

(9) 충돌상태 : Target Object와 Collider Object 간의 거리가 단계별로 설정된 값 이하일 경우 두 Object 간의 상태를 주위/위험/충돌로 표시한다.

(10) 충돌 거리 설정 : 충돌상태의 단계별 간격을 입력하여 충돌상태를 설정한다.

다음은 충돌실험 시뮬레이션의 충돌실험 진행을 제어하는 기능은 Fig. 4를 참조하여 설명하면 다음과 같다.

Fig. 4. Control panel of collision experiment

(1) Auto Reset : Collision Condition Controller의 충돌체 설정값으로 자동 실험의 반복 여부를 선택하여 실험의 반복적인 진행의 편의를 준다.

(2) Repeat : Check Box를 선택하면 설정된 횟수만큼 반복적으로 충돌실험이 진행된다.

(3) Stop 버튼 : 충돌실험의 진행을 정지시키고 실험을 초기화한다.

(4) Play/Pause 버튼 : 충돌실험의 진행 시작과 실험 진행 중에는 실험 일시 정지를 한다.

(5) 진행 스크롤바 : 충돌실험의 진행 과정을 스크롤 기능을 이용하여 조정 확인할 수 있다.

충돌실험 시뮬레이션을 이용한 실험 방법

Unity 3D 엔진으로 개발된 충돌실험 시뮬레이션은 Fixedupdate 함수의 호출 시간, 다양한 형태의 충돌 오브젝트와 충돌체의 설정된 이동속도와 방향으로 Target Object와 Collider Object 간의 충돌을 실시간 검사한다. 충돌체 간의 설정된 가상의 공간상에서 정해진 횟수만큼 반복적으로 충돌실험하고 충돌 시 충돌체 간의 거리를 측정하여 충돌 거리오차의 평균값을 계산하여 오차를 분석하고 정의한다. 또한, 충돌실험을 통해 Unity 3D 엔진의 시간 보정이 발생하고 그 결과 디지털 트윈으로 연동되어 제어되는 장비의 이동 및 위치 제어에 오차가 발생함을 입증한다.

연구결과

실험환경

본 실험에 사용된 PC 환경은 Table 2와 같다.

Table 2. Hardware and software used in the「Collision experiment simulation」

Components	Name and Version
CPU	Intel(R) Core(TM) i7-9700
Mainboard	TUF B365M-PLUS GAMING
BIOS	American Megatrend Inc.
RAM	DDR4 32GBytes
VGA Card	NVIDIA GeForce RTX 2080
OS	Windows 10 Pro
Unity	2019.4.11.f1 Personal

Unity 3D 엔진의 내부에서 사용되는 단위(Unit)를 정의하면 가상의 공간상에 크기나 거리 단위에 1은 실제 1m를 의미한다. 그리고 Fixedupdate 함수는 Time Interval을 설정하여 초당 호출 횟수(FPS : Frame Per Second)를 제어한다. Fixedupdate 함수의 기본 Time Interval은 0.02초로 50 FPS이다.

충돌실험 결과

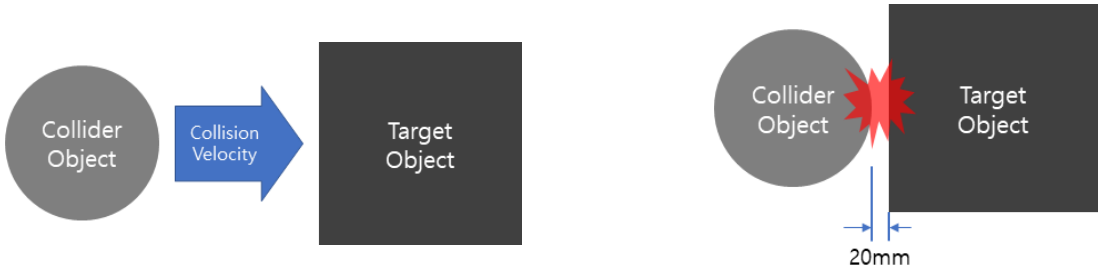
충돌실험 시뮬레이션을 이용한 Unity 3D 엔진에서의 오브젝트 간에 충돌을 감지하는 거리를 측정하기 위해 다음과 같은

실험을 진행하였다. 충돌실험 시뮬레이션에 설정할 수 있는 최저 속도인 1mm/s를 충돌 이동속도로 설정하고 초당 충돌 체크 횟수를 50 FPS부터 50 FPS 단위로 증가시켜 200 FPS까지 충돌실험을 진행하였다. 각각의 실험 조건에 따라 100회 실행하여 오브젝트 간의 충돌 간격 평균값을 Table 3과 같다.

Table 3. Collision distance according to FPS(Frame Per Second) at minimum speed

FPS	Velocity(mm/s)	Collision Distance(mm)	Error Distance(mm)
50	1	19.9886	0.0114
100	1	19.9938	0.0062
150	1	19.9959	0.0041
200	1	19.9972	0.0028

위 Table 3의 결과를 토대로 오브젝트 간의 충돌 인식 거리를 유추해보면 단위 시간당 많은 횟수로 충돌 체크를 할수록 오차 간격은 0.0114mm에서 0.0028mm까지 줄어드는 것으로 확인되고 충돌 감지 거리는 20mm에 가까워진다. 이를 통해 Unity 3D 엔진에서 오브젝트 간의 거리가 20mm인 경우 충돌이 발생한 것으로 추측할 수 있다. 위 결과를 토대로 이하 실험에서는 오브젝트 간의 충돌은 Fig. 5와 같이 20mm 접근을 기준으로 하고 이하의 오차 거리는 시간 오차로 인해 충돌 거리 오차가 발생한 것으로 가정한다.



<Before collision> <After collision>

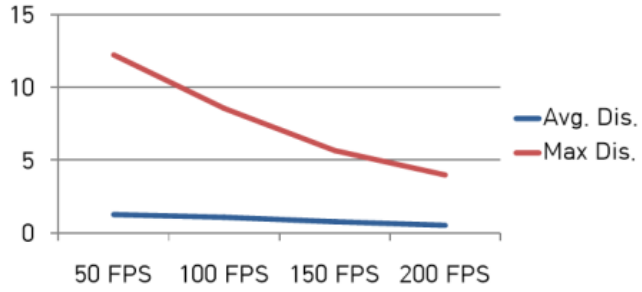
Fig. 5. Object collision status of unity 3D engine

위 실험을 근거로 식 (1)을 도출할 수 있다.

$$\text{오차 간격} = \text{충돌 감지 간격} - \text{충돌 오차 평균값} \quad (1)$$

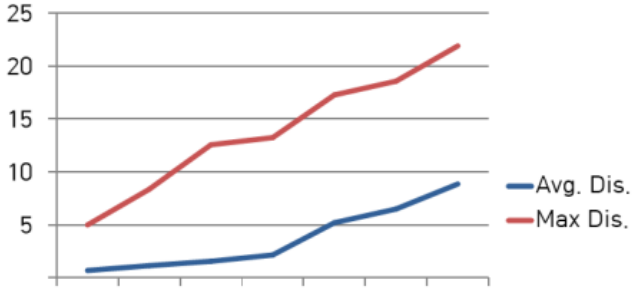
식 (1)에서의 충돌 감지 간격은 Unity 3D 엔진에서 오브젝트 간의 충돌을 인지하는 거리인 20mm이고 충돌 오차 평균값은 충돌실험 시뮬레이션을 통해 도출한 실험값이다.

다음 충돌실험은 이동속도 200mm/s 상에서 FPS에 따른 충돌실험 100회 오차 간격 평균값과 최대 오차 상위 5%의 평균값은 Fig. 6과 같다. 하드웨어의 성능과 모델링 데이터에 따라 FPS의 값은 클수록 오차 간격은 최소화할 수 있다.



<Error distance according to FPS at collision velocity 200mm/sec>

Fig. 6. Curved line of Error Distance-FPS



<Error distance according to collision velocity>

Fig. 7. Curved line of Error Distance-Collision Velocity

100 FPS 상에서 충돌 오브젝트의 이동속도에 따른 오차 간격은 설정된 이동속도별 충돌실험을 100회 진행하여 산출된 거리오차의 평균값과 상위 5%의 최대 오차 평균값은 도식화하면 Fig. 7과 같다.

충돌실험 결과 분석

Unity 3D 엔진을 기반으로 한 충돌실험 시뮬레이션을 이용해 실험한 결과를 보면 테스트 환경(운영PC 성능 및 모델링의 크기 등)을 고려해도 수 millisecond의 시간 오차 보정이 확인된다. 이로 인해 충돌체의 이동제어에 단위 시간당 거리오차가 발생하고 거리오차는 충돌체의 이동속도와 비례한다. 이를 통해 디지털 트윈을 이용한 원격해체 시뮬레이터는 하드웨어와 소프트웨어 환경과 수동 제어 시 정밀 제어 장치의 요구 정밀도에 따른 이동속도의 제한이 필요할 것으로 판단된다.

결 론

충돌실험 결과를 토대로 디지털 트윈이 적용된 원격해체 시뮬레이터를 이용한 장치 제어 시 시간 오차 보정에 따른 정밀 제어 장치의 최대 이동속도 제한이 필요하다. 제한조건은 정밀도 10mm 이내의 오차를 기준으로 한다면 다음과 같다. 예로 디지털 트윈 기술을 이용하여 장치의 실시간 정밀 제어를 하는 경우 대상 물체와의 허용오차 간격을 정의하여 기준으로 설정한다면 다음과 같을 것이다.

Table 4. Permission error distance according to control velocity

Approach Distance(mm)	Control Velocity(mm/sec)	Permission Error Distance(mm)
500 ~	1200	20
200 ~ 500	500	15
100 ~ 200	200	10
~ 100	100	5

Table 4와 같은 근접거리에 따른 제어 장비의 가능 속도를 구하고 이를 통해 허용 거리오차를 고려하여 정밀 제어를 한다면 요구되는 정밀도의 신뢰도는 충분히 만족시킬 수 있다.

Unity 3D 엔진에 Fixedupdate 함수의 1/1000초(Milli Secend) 단위의 시간 보정으로 인해 정밀 제어 시 발생하는 오차를 고려하여 디지털 트윈을 적용한 시뮬레이션 프로그램 개발 기획 및 설계가 이루어져야 한다. 또한 운용 제어 장비의 가용 및 허용 오차와 작업의 요구 속도를 시스템 개발 환경, 하드웨어 사양과 시뮬레이션에 모사된 제어 장비 및 시설물 등의 모델링 데이터의 크기도 반드시 고려한다.

Acknowledgement

이 논문은 2020년 한국연구재단 ICT기반원자력안전혁신기술개발사업의 지원을 받아 수행하였습니다.

References

[1] Chabal, C., Mante, J.F., Idasiak, J.M. (2011). “Virtual reality technologies: A way to verify and design dismantling operations, first application case in a highly radioactive cell.” International Journal on Advances in Intelligent Systems, Vol. 4, No. 3&4, pp. 343-356.

[2] Choi, H.-W., Kang, S.-M., Kim, K.-J., Kim, D.-Y., Choung, Y.-J. (2015). “Development of the visualization prototype of radar rainfall data using the unity 3D engine.” Journal of the Korean Association of Geographic Information Studies, Vol. 18, No. 4, pp. 131-144.

[3] Fan, W., Lee, C.-H., Chen, J.-H. (2016). “Real-time repairable interpolation scheme for CNC tool path processing.” International Journal of Precision Engineering and Manufacturing, Vol. 17, No. 12, pp. 1673-1684.

[4] Kim, H.J., Kim, H.-S, Kim, W.-T. (2018). “The human digital twin architecture for Human-In-the-Loop system.” Proceedings of Symposium of the Korean Institute of Communications and Information Sciences, 2018, Vol. 1, pp. 631-632.

[5] Kim, M.-S., Sung, N.-J., Choi, Y.-J., Hong, M. (2017). “Performance comparison of particle simulation using GPU between OpenGL and unity.” KIPS transactions. Software and data engineering, Vol. 6, No. 10, pp. 479-486.

[6] Kim, M.-S., Song, W., Choi, Y.-J., Hong, M. (2018). “Real-time collision response between cloth and sphere object in unity.” Journal of Internet Computing and Services, Vol. 19, No. 6, pp. 53-62.

[7] Lee, E.-h., Kim, C.-L. (2019). “Radiological impact on decommissioning workers of operating multi-unit NPP.” Journal of Nuclear Fuel Cycle and Waste Technology(JNFCWT), Vol. 17, No. 1, pp. 107-120.

[8] Lee, K.-S. (2019). “A study of spiderwebs via unity physics engine.” School Science Journal, Vol. 13, No. 3, pp. 261-270.

[9] Song, S.-M., Lee, S.-J., Jeon, J.-W. (2013). “Real-time correction system for improving the disparity image.” 13th International Conference on Control, Automation and Systems (ICCAS 2013). pp. 1138-1142.