JDCS
디지털콘텐츠학회논문지
Journal of Digital Contents Society
Vol. 22, No. 10, pp. 1715-1719, Oct. 2021

Check for updates

최적화된 멀티 스레드 프로그래밍 연구 - Unity DOTS에 대한 정보와 활용

송 현 철
남서울대학교 가상증강현실융합학과 조교수

A Study on Optimized Multi-Thread Porgramming - Information and The Use of Unitiy DOTS

Hyun-Chul Song

Assistant Professor, Department of Virtual Reality and Augmented Reality, Nam-Seoul University, Cheonan 31020, Korea

[요 약]

최근 게임 개발자들이 게임 엔진을 활용하여 콘텐츠를 개발하다 보면 가장 애쓰는 부분이 속도와 성능이다. 특히 큰 규모의 콘텐츠의 경우 수많은 최적화를 필요로 한다. 더불어 게임 소비자가 2개 이상의 하드웨어 코어를 가지게 된지 10년이 넘은 지금까지도 멀티 코어를 효과적으로 사용하는 프로그램을 제공하기란 매우 어려운 상태다. Unity는 고성능 멀티스레드 데이터 지향 기술 스택(DOTS; Data-Oriented Technology Stack)으로 엔진의 핵심 기반을 재구축하고 있다. 본 연구는 DOTS에 포함된 기능(C# Job System, Entity Componet System, Burst Compiler)을 활용 시 프로그래머는 가이드라인만 맞추어 코딩하면 성능을 상당히 향상시키면서 쉽게 멀티스레드 코드를 작성할 수 있다는 것을 알 수 있고 오늘날 복잡하거나 대규모의 콘텐츠 제작에 있어서 적절한 최적화된 멀티 스레드 프로그래밍 기법을 제안하는 것에 목적이 있다.

[Abstract]

As game developers use game engines to develop content recently, the most difficult part is speed and performance. A number of optimizations are required, especially for large-scale content. In addition, it is still very difficult to provide a program that effectively uses multiple cores, even after more than 10 years of having more than two hardware cores for game consumers. Unity is rebuilding the core base of the engine with its high-performance, multi-thread data-oriented technology stack (DOTS). The purpose of this study is to use the functions(C# Job System, Entity Component System, Burst Compiler)included in DOTS to show that programmers can write multi-thread code easily while significantly improving performance by only coding according to guidelines, and to suggest appropriate optimized multi thread programming techniques for today's complex or large-scale content production.

색인어 : 최적화, 멀티 스레드 프로그래밍, 데이터 지향 설계, 데이터 지향 기술 스텍

Keyword : Optimization, multi thread programming, Data-Oriented Design, Unity DOTS

http://dx.doi.org/10.9728/dcs.2021.22.11.1715

Received 02 September 2021; **Revised** 14 September 2021
Accepted 25 October 2021

***Corresponding Author; Hyun-Chul Song**

Tel: +82-41-580-2703
E-mail: hcsong@nsu.ac.kr

Ⅰ. 서 론

최근 가장 큰 화제를 모은 게임을 꼽자면 단연 '포켓몬 고'를 빼놓을 수 없다. '포켓몬 고'를 실행하면 게임 첫 화면에 'UNITY 3D'라는 로고를 볼 수 있다. 게임 뒷단에 '유니티'라는 기술이 사용됐다는 의미다. '포켓몬 고'뿐만 아니다. 유니티는 현재 모바일 게임 분야에서 널리 사용되고 있다. 유니티의 슬로건도 '게임 개발의 민주화(democratize game development)'다. 누구나 쉽게 게임을 만들 수 있게 도와 게임 업계를 성장시키겠다는 포부인 셈이다. 실제로 유니티는 모바일 게임 개발자에게는 없어서는 안 될 존재로 성장했다[1].

유니티는 게임 엔진 기술이자 통합개발환경(Integrated Development Environment, IDE)이다. 연구자 역시, 3D게임이나 가상현실(VR)과 증강현실(AR)관련 콘텐츠를 제작할 때, 유니티 엔진을 이용해 제작하게 되었다[2]. 제작을 시작할 때는 쉬워 보이기도 했으나 수많은 객체(Object)들을 생성했을 때 모든 물리연산이 싱글 스레드로 돌아간다는 큰 문제점 때문에 최적화에 애먹기도 했다. 유니티 엔진 특성상 하나의 스레드가 각각의 Object에 내제된 물리엔진을 포함한 기타 로직들을 하나씩 순회하면서 연산하다보니 Object가 1000~2000개만 넘어가도 일시적으로 멈춤현상이 발생하면서 원하는 퍼포먼스가 나오지 않는 결과를 불러왔다.

하지만 현재 유니티에서 사용되는 구조인 Mono Behaviour 기반의 기술들이 점점 DOTS기반의 기술로 전파되고 있으며 내부 인터널 엔진들(ex. Job System) 또한 도입하여 바뀌어가고 있다[3]. 본 연구의 목적은 바뀌어가고 있는 유니티 기반 기술인 DOTS를 활용한 최적화된 멀티 스레드 프로그래밍 기법을 제안하는 것에 있다.

Ⅱ. 관 련 연 구

2-1 DOTS란?

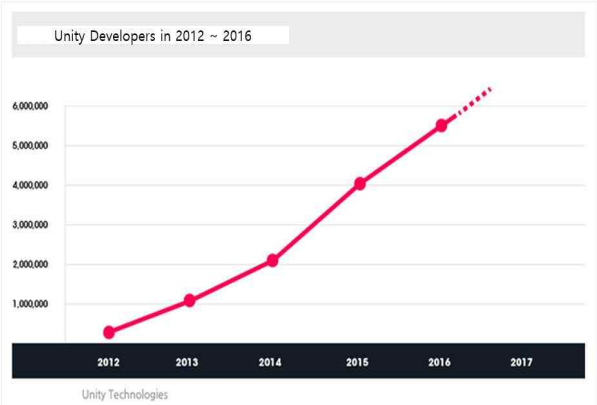

그림 1. 'Unity 등록 개발자 수' <unity.com>
Fig. 1. 'Unity Registered Developers Counts'

현재 유니티는 복잡한 프로그래밍으로 고생할 필요 없이 게임에 최신 멀티코어 프로세서를 완전히 활용할 수 있도록 유니티의 핵심 기반을 고성능 멀티스레드 데이터 지향 기술 스택(DOTS)으로 재구축하였다. DOTS는 'Data-Oriented Technology Stack'의 약자이며 기본적으로 ECS라는 데이터(Component) 관리 방식과 Job System이란 함수지향적 디자인을 택한 연산방식을 통하여 개발자로 하여금 가이드라인만 맞추어 코딩하면 코드를 여러 프로젝트에서 더 쉽게 읽고 재사용할 수 있게 하였다. 여기에 어떠한 코드 추가 작업 없이 더 빠른 성능을 가져다주는 새로운 컴파일러인 Burst Compiler를 추가하여 CPU부하가 심한 작업의 경우 약 20배 가까운 퍼포먼스 향상을 가져와 주도록 하였다.

2-2 멀티스레드(Multi-Trhead)

본멀티 스레딩은 CPU 내 여러 코어에서 한 번에 여러 개의 스레드를 처리하는 CPU성능을 활용하는 프로그래밍의 한 유형으로 한 번에 하나가 아니라, 동시에 여러 개의 작업 또는 명령을 실행한다[4]. 멀티스레드 시스템을 활용 시 다양한 하드웨어에서 실행 가능한 게임 제작이 가능하고 더 많은 요소와 더 복잡한 시뮬레이션이 있는 풍부한 게임 월드 구축, 모바일 기기의 과열과 배터리 수명이 단축되지 않도록 최적화 또한 가능하다[5]. 유니티 DOTS에서 제시하는 Guideline에 따라 멀티 스레드 코드를 작성하면 코드를 더욱 간편하게 재사용하고 다른 작업자가 코드를 더욱 간편하게 이해하고 협업할 수 있게 된다.

2-3 Data-Oriented

DOTS의 Data-Oriented는 데이터 지향이란 뜻이다. 컴퓨터 프로그래밍론을 떠올리고자 하면 절차지향과 객체지향만 생각하면 끝났던 이전날과 다르게 오늘날에는 함수지향(Functional Programming)개념의 부활에 뒤따라 함수반응형프로그래밍(Functional Reactive Programming)가 생겼고 이외에도 MVC(Model View Controller)패턴과 이에 따라 MVVM(Model-View-ViewModel), MVP(Model View Presenter)와 같은 개념도 추가되었으며 또 하나가 추가된 것이 바로 Data-Oriented Design(데이터지향 설계)이다. 본 연구에서는 유니티 엔진에서의 변천과정에 포함된 객체 지향 디자인과 데이터 지향 디자인에 대하여 다루어 보겠다.

객체 지향 디자인(이하 객체지향)은 객체 중심(Object, 단순 데이터가 아니라 그 데이터의 조작방법에 대한 정보도 포함된 것), 객체를 대상으로 다루는 수법이다[6]. 그리고 데이터 지향 디자인(이하 데이터지향)은 데이터 중심, 데이터 우선으로 데이터를 구조화 시키는 수법이다[7]. 기존에 유니티 엔진의 큰 핵심 기반이었던 객체 지향에서 데이터 지향으로 전환하면서 어떠한 효과적인 결과를 낳았는지 비교분석 해보았다.

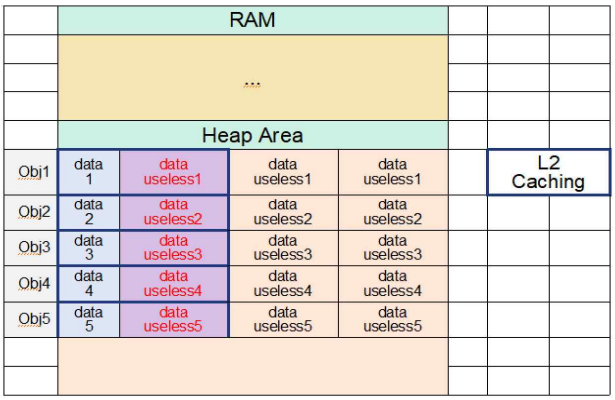

그림 2. 객체지향의 경우 메모리영역
Fig. 2. Memory area for object-orientation

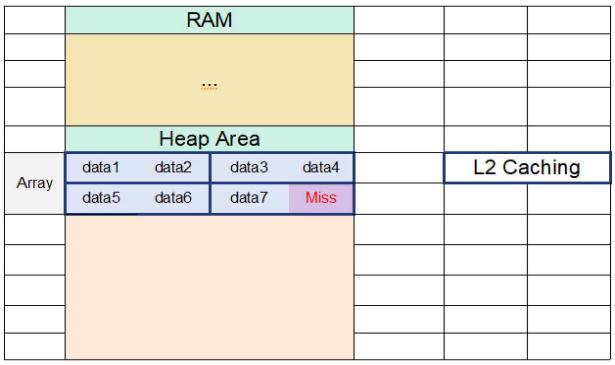

그림 3. 데이터지향의 경우 메모리영역
Fig. 3. Memory area for data-orientation

먼저 객체지향의 경우 일반적으로 모든 연산해야 할 데이터가 객체 속에 존재하고 각 객체는 RAM의 Heap영역에서 자리가 비워지면 들어가 저장한다. 그 결과는 다음 그림 2와 같다.

각 객체가 포함하고 있는 데이터는 연속적으로 저장되어 있기 때문에, 한 번에 두 칸씩 Caching을 한다고 할 경우 연산에 필요한 데이터 1부터 5까지의 정보를 꺼내오기까지 총 5번의 Cache Miss가 발생하게 된다.

반면 데이터지향의 경우 객체지향에서 각 객체에 들어갈 데이터들을 종류별로 한데 모아 하나의 Array에 저장한다. 이러한 경우 다음 그림 3과 같이 연산해야 할 데이터는 RAM 안에서 모두 연속된 공간에 존재하게 된다.

단순히 저장방법만 바꿨을 뿐인데 객체지향에서는 5번의 캐싱으로 5개의 데이터를 불러올 동안 데이터지향에서는 4번의 캐싱으로 총 7개의 데이터를 불러올 수 있게 된다.

이렇듯 데이터지향이 Cache Miss를 줄여 더 빠른 연산속도를 끌어낸다는 것을 이론적으론 쉽게 알 수 있지만 구체적으로 이것이 얼마나 큰 효과를 발휘하는지에 대해서는 이후 내용 중 사례적용 부분에서 실제 project에 적용한 모습과 함께 FrameWork 분석을 통해 보여주겠다.

2-4 Entity Component System(ECS)

ECS는 유니티의 컴포넌트 시스템을 데이터 지향적인 접근 방식을 이용하여 새롭게 만든 시스템으로 놀라운 성능향상과 병렬성을 보여준다. ECS는 해결하려는 문제, 즉 게임을 구성하는 데이터와 동작에 중점을 두는 코드 작성 방식이며 ECS를 사용하면 디자인상으로 게임 프로그래밍에 대한 우수한 접근 방식일 뿐만 아니라 DOTS의 나머지 두 기능(C# Job System과 Burst Compiler)을 활용하여 오늘날의 멀티코어 프로세서를 최대한 활용할 수 있는 이상적인 포지션을 차지할 수 있고 객체 지향에서 데이터 지향으로 전환하여 코드를 더욱 간편하게 재사용할 수 있다.

2-5 C# Job System

C# Job System은 C++ Job System을 C#, 사용자 스크립트까지 확장시켜 유니티 내부 컴포넌트와 함께 잡 시스템을 지원하도록 하였으며 오늘날 컴퓨터의 멀티코어를 활용한다. 사용자가 안전하고, 빠르고, 정교한 코드를 작성하면서 경쟁상태와 같은 멀테스레딩의 취약점을 방지하고 멀티스레드 코드를 효율적으로 실행한다[4].

2-6 Burst Compiler

Burst Compiler는 HPC#(High Performance)과 더불어 유니티가 지향한 low-level 기반 기술로 새로운 백 엔드 컴파일러 기술을 통해 C# 잡을 수행하고 고도로 최적화된 machine code를 생성한다. 번거로운 수고 없이 여러 개의 플랫폼에 걸쳐 수작업으로 조정한 어셈블러 코드의 다양한 이점을 얻을 수 있으며, 이는 누구나 이 스택으로 게임엔진을 만들 수 있게 하기 위함이다[8].

Ⅲ. 사례 연구와 적용

3-1 DOTS 환경구축

유니티 DOTS는 현재 Unity Editor 2019 버전 1.0부터 preview package로 제공된다. Unity Editor에서 Window 탭 – Package Manager 실행 후 Burst, Entities, Hybrid Renderer 이 세가지 패키지를 설치하면 DOTS를 활용할 수 있는 환경이 구축된다.

3-2 큐브 50만개 생성 예제와 FrameWork 비교분석

그림 4. 'Unity DOTS' 패키지 다운로드 절차
Fig. 4. 'Unity DOTS' Package download procedure

본 연구자는 유니티 DOTS를 활용하여 큐브 오브젝트를 Entity화하여 50만개의 큐브를 생성하고 모든 큐브가 제자리에서 회전하는 프로젝트를 만들었다. 엔티티 시스템의 Convert to Entity 스크립트와 가이드를 활용해 큐브 오브젝트를 Entity로 컨버팅하였고 Job System 또한 가이드라인에 맞춰 코딩하여 생성뿐만 아니라 회전까지 하도록 하고 Burst Compiler를 활용하여 성공적으로 예제를 완성하였다. 그림 5와 6은 해당 프로젝트의 FrameWork 분석 및 비교 자료이다.

DOTS 활용 전에는 프레임이 15fps 아래로 떨어진 채 유지되는 것을 그림 5을 통해 확인할 수 있다.

DOTS 활용 시에는 프레임이 200fps 이상으로 유지되는 것을 그림 6을 통해 확인할 수 있다.

표 1은 해당 예제의 Built-In(DOTS적용 전)과 DOTS적용 후의 FrameWork 비교분석 그래프이다.

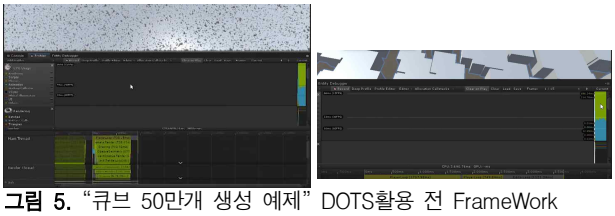

그림 5. "큐브 50만개 생성 예제" DOTS활용 전 FrameWork
Fig. 5. "Example of creating 500,000 cubes" FrameWork Before DOTS Utilization

그림 6. "큐브 50만개 생성 예제" DOTS활용 후 FrameWork
Fig. 6. "Example of creating 500,000 cubes" FrameWork after DOTS Utilization

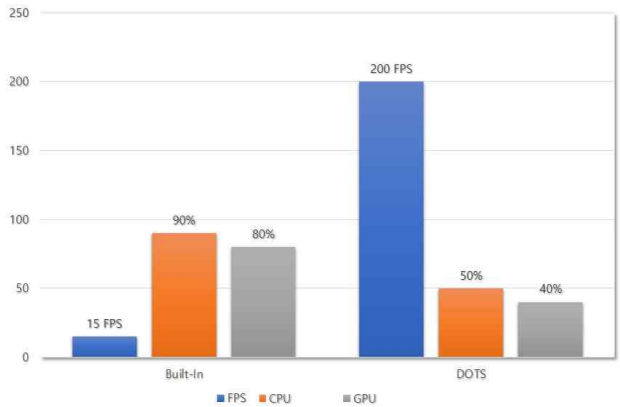

그림 7. "큐브 50만개 생성 예제" DOTS 비교분석
Fig. 7. "Example of creating 500,000 cubes" FrameWork Comparative Analysis

3-3 실제 게임 콘텐츠에 적용과 FrameWork 비교분석

큐브 50만개 프로젝트에 이어 실제 게임 콘텐츠에도 DOTS를 적용해 보았다. 유니티에서 제공되는 우주 컨셉의 3인칭 슈팅 게임으로 플레이어는 'Spread Shot'이라는 스킬로 이 스킬이 활성화되면 0.1초당 한번에 400발의 총알을 발사할 수 있게 된다. 이 총알 오브젝트를 한번에 생성할 시 Frame Drop과 함께 화면이 일시정지되는 현상을 피할 수 없다. 하지만 총알 오브젝트를 Entity화하고 잡 시스템을 활용하여 총알의 position이동, rotation값을 부여하는 Job을 생성했고 위 예제와 마찬가지로 Burst Compile도 성공적으로 마쳤다. 그림 5와 6는 해당 프로젝트의 FrameWork 분석 및 비교 자료이다.

DOTS 적용 전에는 총알 오브젝트의 클론들이 생성되는 것을 Hierarchy창에서 확인할 수 있으며 아래 Profiler 그래프를 통해 프레임이 15fps 아래로 떨어지며 화면이 끊기는 현상이 있다. 반면에 DOTS를 적용한 후에는 총알 오브젝트들이 정상적으로 Entity화 하였고 프레임 또한 60~100fps로 유지되며 Spread Shot 스킬이 화면끊김 없이 성공적으로 실행되는 것을 확인할 수 있다.

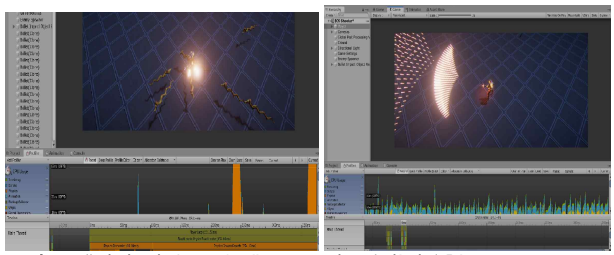

그림 8. "실제 게임 콘텐츠" DOTS적용 '전'과 '후' FrameWork
Fig. 8. "Real Game Content" DOTS Application 'Before' and 'After' FrameWork

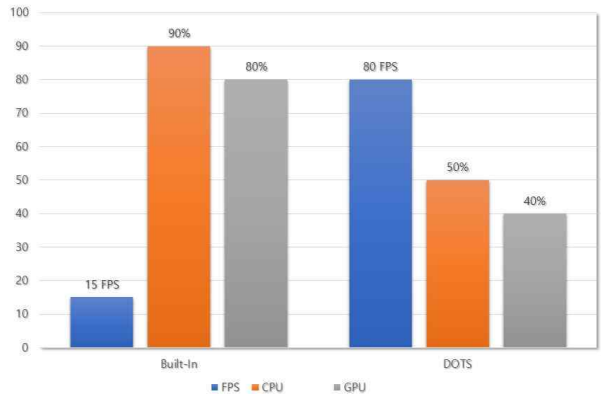

그림 9. 실제 게임 콘텐츠에 적용 '전'과 '후' FrameWork 비교분석
Fig. 9. Comparative analysis of 'before' and 'after' FrameWork applied to real game content

Ⅳ. 결 론

유니티 DOTS는 3D 게임이나 3D콘텐츠 개발에 있어서 성능과 속도를 최고로 높이는 최적화에 매우 필요한 기법이다. 작년 초부터 이런 기법들은 연구가 이뤄졌으며 Unity 2019 버전부터 Preview Package로 제공되고 있어 쉽게 구현할 수 있게 되었다. 프로젝트 내 오브젝트들을 Entity로 컨버팅하고 C# Job System으로 멀티스레드 코딩을 하여 잡을 수행시키고 Burst Compiler로 컴파일링하여 보다 최적화된 프로젝트로 완성시켰다. 한 Scene에 50만개의 큐브를 생성하고 회전시켰으며 0.1초 당 400발의 총알을 생성해 이동시키는 데에 이상 없이 60fps 또는 그 이상으로 높은 속도와 성능을 확인하였다. 이 수치는 객체 수나 수행시킬 잡의 수에 따라 달라질 수 있지만, 그 수가 더 늘어나면 늘어날수록 DOTS 기술은 더더욱 빛을 보여줄 것이다.

본 연구를 통해 얻게 된 결과는 연구자의 멀티스레드 프로그래밍과 VR·AR 3D콘텐츠 제작에 연구 자료로 사용될 것이며 앞으로 DOTS뿐 아니라 Unity Physics나 Havok Physics와 같은 물리 엔진에 대해서도 수치를 분석하고 기술을 활용하는 자료를 제안하여 프로그래밍과 3D콘텐츠 제작에 도움이 되는 연구를 하고자 한다.

감사의 글

이 논문은 2020년도 남서울대학교 학술연구비 지원에 의해 연구되었음

참고문헌

[1] Haas J. K., A history of the Unity game engine, Worcester Polytechnic Institute, Worcester, England, E-project-030614-143124, pp.1-44, 2014

[2] Linowes J., *Unity virtual reality projects*, Packt Publishing Ltd, 2015.

[3] Xie, J. “Research on key technologies base Unity3D game engine,” in *Proceeding of the 7th international conference on computer science & education*, Melbourne, Australia, pp. 695-699, July 2012. https://doi.org/10.1109/ICCSE.2012.6295169

[4] Berger, E. D., Yang, T., Liu, T. and Novark G., “Grace: safe multithreaded programming for C/C++,” in *Proceedings of the 24th ACM SIGPLAN conference on Object oriented programming systems languages and applications,* Orlando, Florida, USA, pp. 81-96, October 2009. https://doi.org/10.1145/1640089.1640096

[5] Jie, J., Yang, K., and Haihui, S., “Research on the 3D game scene optimization of mobile phone based on the Unity 3D engine,” in *Proceedings of the 2011 International Conference on Computational and Information Sciences,* Chengdu, China, pp. 875-877, October 2011. https://doi.org/10.1109/ICCIS.2011.317

[6] Cox, Brad J., *Object oriented programming: an evolutionary approach,* Addison-Wesley Longman Publishing Co., Inc., 1986.

[7] Hu, H., Shinde, S., Adrian, S., Chua, Z. L., Saxena, P. and Liang, Z., “Data-oriented programming: On the expressiveness of non-control data attacks,” in *Proceedings of the 2016 IEEE Symposium on Security and Privac*y, San Jose, CA, USA, pp. 969-986, May 2016. https://doi.org/10.1109/SP.2016.62

[8] Joon-Kee Choi, “A Study on a Performance Enhancement Technique of Optimizing Compiler,” Computer Iniformation Society of Korea, Vol. 5, No. 1, pp. 17-27, February 2000.

송현철(Hyun-Chul Song)

2005년 : 중앙대학교 컴퓨터공학과 대학원 (공학석사)
2019년 : 중앙대학교 소프트웨어학부 (공학박사-컴퓨터공학과)

2009년～2012년: 박사과정 전문연구요원
2014년～2017년: 남서울대학교 멀티미디어학과 외래교수
2018년～2020년: 남서울대학교 가상증강현실대학원 연구교수
2020년～현 재: 남서울대학교 가상증강현실대학원 전임교수
※관심분야 : 컴퓨터비전(Computer Vision), 인공지능(AI), 딥러닝(Deep Learning), 증강현실() 등