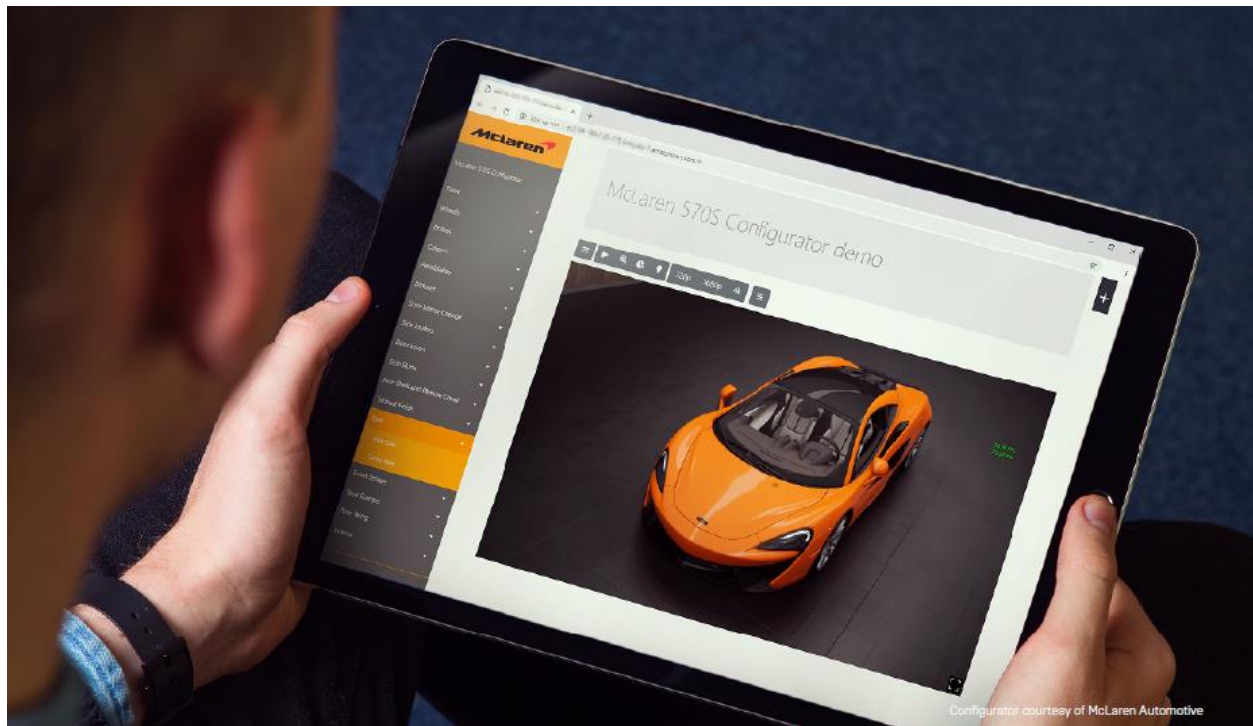


언리얼 엔진 콘텐츠의 멀티 플랫폼 스트리밍

HTML5, WebGL, 픽셀 스트리밍 비교



목차

목차	2
도입	3
콘텐츠 배포의 어려운 점	4
UX 및 퍼포먼스 고려사항	5
기술적 고려사항	5
배포 솔루션 비교	6
WebGL	6
WebGL 저작도구	7
WebGL 제약사항	7
HTML5	8
HTML5 제약 사항	9
픽셀 스트리밍	10
WebRTC 프로토콜	11
요구 사항	11
비디오 카드	11
픽셀 스트리밍 사용 사례	13
LAN 내 협업	13
커뮤니케이션, 승인, 프레젠테이션	14
원격 협업 디자인, 공유 경험, 소비자 애플리케이션, 컨피규레이터	15
요약	17
심화 단계	17
본 문서 소개	19

저자	19
기여자	19
편집	19

도입

연결된 사용자 경험 개발 과정에서 콘텐츠 공유는 언제나 협업, 제작, 퍼블리싱에서의 핵심 의사결정 대상입니다. 사용자가 PC, 태블릿, 스마트폰 등의 다양한 디바이스를 통해 공유 콘텐츠와 상호작용하고 이를 소비할 때 다음과 같은 핵심적인 의문점이 나타납니다. 다양한 플랫폼과 하드웨어 성능 환경을 대상으로 어떻게 퍼블리시하는 것이 좋을까요?

흔한 접근 방법은 가장 낮은 진입 지점과 타깃 사용자가 수용할 수 있는 퀄리티의 수준을 찾아 이를 기준 플랫폼으로 활용하는 것입니다. 하지만 이런 접근 방법은 모든 최종 사용자 경험의 퀄리티를 제한합니다.

또한 이러한 디플로이먼트에서는 최종 사용자가 실시간 콘텐츠를 소비할 때 디바이스는 데이터와 로직을 담고 있고 그 결과를 화면에 렌더하는 디바이스와 같은 디바이스입니다. 디바이스에 데이터를 다운로드하다 보면 몇가지 문제가 발생합니다. 예를 들면, 빠른 다운로드 속도와 더 적은 설치 용량을 위해 퀄리티를 제한해야 하는 문제가 있습니다.

이 문서는 픽셀 스트리밍으로 언리얼 엔진 기반 경험을 공유할 때 주어지는 옵션과 툴을 다른 옵션과 비교 분석합니다. 목표는 언리얼 엔진의 게임 외 애플리케이션 경험의 퀄리티와 기능을 가장 높은 수준으로 유지하는 것입니다.



콘텐츠 배포의 어려운 점

정보의 시대에서 경험의 시대로 발전하면서 소비자는 참여형, 몰입형 고퀄리티 콘텐츠에 가장 많은 관심과 반응을 보입니다. AR 또는 XR 오버레이와 같은 3D 모델 기반 상호작용 경험을 담은 콘텐츠가 계속 늘어나고 있습니다. 최대한 많은 소비자를 확보하려면 제작사는 휴대폰, 태블릿, PC, 인터랙티브 디스플레이 등의 다양한 플랫폼에 콘텐츠를 공유해야 합니다.

WebGL 또는 HTML5 에 기반한 과거 또는 현재 솔루션으로는 디스플레이 콘텐츠의 충실도와 상호작용성이 소비자의 디바이스에 따라 달라집니다. 특히 하드웨어, 디스플레이 메커니즘, 운영체제에 따라 달라집니다. 즉, 콘텐츠 사용자를 최대한 확보하려면 가장 하위 수준 디바이스를 공유 애플리케이션 기준으로 정하거나 애플리케이션을 다양한 버전으로 제작해 서로 다른 사용자 그룹을 지원해야 한다는 것을 의미합니다.

스트리밍 솔루션 개발의 어려운 점은 콘텐츠를 여러 커뮤니케이션 채널로 전달하면서, 최종 사용자의 디바이스에 상관없이 브랜드 이미지에 따른 룩앤필, 충실도, 상호작용형 기능을 유지하는 것입니다. 이 문제는 공유 애플리케이션과 고성능 하드웨어 요구 사양을 사용자의 디스플레이 디바이스와 분리해 해결할 수 있습니다.

UX 및 퍼포먼스 고려사항

다양한 사용자 디바이스에 공유 애플리케이션을 배포하려면 사용자 경험과 이를 구현하는 기술에서 다양한 요소를 고려해야 합니다.

상호작용 - 디플로이가 성공적이려면 브라우저 내 사용자 입력과 호스트 반응 사이의 시간 간격이 일정 시간 내여야 사용자가 그 경험을 즐길 수 있습니다. 이는 최초 실행을 위한 대기 시간을 포함합니다. 일반적으로 게임 외의 콘텐츠 소비자는 속도가 빠른 게임 소비자보다 인내심 한계치가 더 높습니다. 자동차 컨피규레이터를 예로 들면, 사용자는 어떠한 반응을 보기까지 1~2 초 정도 기다리고 최종 퀄리티를 보기까지 10 초 정도 기다립니다.

재생 속도 - 콘텐츠는 실시간 퍼포먼스에 최적화되어야 합니다. 먼저 특정 애플리케이션과 그 이용자가 수용할 수 있는 재생 속도가 어느 정도인지 정해야 합니다. 일반적으로 사람의 몰입형 인식(Immersive perception) 수용 범위는 60fps 이지만, 5fps 만 요구하는 애플리케이션도 있으며 90fps 까지 요구하기도 합니다.

이미지 퀄리티 - 사용자가 보고 경험하는 퀄리티와 충실도 수준은 애플리케이션 선에서 콘텐츠 구성(실시간 재생 포함), 인코딩 퀄리티, 하드웨어의 GPU 에 따라 정해집니다. 목표 퀄리티와 이에 따른 비용은 사업 내용과 콘텐츠 구성의 최종 출력을 정합니다.

기술적 고려사항

호스팅과 데이터 - 배포된 경험은 클라이언트 또는 애플리케이션 측에서 실행할 수 있습니다. 하지만 클라이언트 플랫폼의 퀄리티 제한, 시간, 저장 공간, 클라이언트로 데이터를 전송하는 방식에 내재한 보안 문제가 발생할 수 있습니다. 그러므로 가장 이상적인 방식은 애플리케이션에서 스트리밍하는 것입니다. 사용자 수가 제한된 단순한 애플리케이션에서는 접근 가능한 단일 워크스테이션에서 스트리밍하는 솔루션으로 충분히 해결될 수도 있습니다. 사용자 수가 많으면서 각 사용자가 특정 버전에 대한 완전한 제어권을 갖는 애플리케이션에서는 클라우드 솔루션이 더 나은 사용성과 확장성을 보장합니다. 사용자 수가 적은 복잡한 애플리케이션도 클라우드 호스팅을 사용하는 것을 권장합니다. 클라우드 솔루션을 사용하면 반드시 저장 공간 비용에 따라 콘텐츠 데이터 크기를 고려해야 합니다.

사용자 로드 - 사용자 수와 애플리케이션 복잡도는 호스트의 단일 인스턴스가 무엇을 제공하는지와 접속 중인 사용자 대응에 어떤 로드 밸런싱이 필요한지 정합니다. 충실도가 높은 완전한 상호작용형 자동차 컨피규레이터에서는 구성이 동일한 인스턴스 10,000 개가 필요할 수 있습니다. 최초 상호작용이 얼마나 빠르게 발생하느냐에 따라 서버 개수가 핵심이 됩니다.

스트림 크기 - 타겟 사용자 그룹이 모바일 디바이스와 같은 특정 플랫폼을 사용하고 있다면, 해당 필드 내 대역폭 제한에 부합하는 비디오 스트림 크기를 구체적으로 정해야 할 수 있습니다.

업데이트 - 업데이트와 버그 픽스를 어떻게 배포해야 할까요? 전체 애플리케이션을 항상 교체해야 할까요? 아니면 실시간 업데이트가 가능할까요? 더 많은 수의 인스턴스들을 대상으로는 어떻게 업데이트와 픽스를 적용할지 고민해야 합니다.

보안 - 호스트와 클라이언트 사이의 접속과 보안이 먼저 사용 사례를 정할 수 있습니다. 디자인 리뷰를 위해 기밀 데이터에 대한 시스템 쿨을 공유하고 논의해야 할 때, 또는 새로운 정보를 특정 시간 내에만 배포하고 그 시간 외에는 접근이 불가능해야 할 때, 네트워크 또는 접속 사이트에 보안 조치를 구현해야 할 수도 있습니다.

메트릭스 - 사용자 행동과 신(Scene) 통계가 KPI 인 경우, 분석 소프트웨어로 추적하고 저장해야 할 이벤트를 정의하고, 활성화하고 연결해야 할 수 있습니다.

배포 솔루션 비교

에픽게임즈가 언리얼 엔진(UE4) 콘텐츠의 배포 솔루션을 제공하기 위해 준비할 때 개발팀은 기존 기술을 솔루션의 중추로 사용하려 했습니다. 후보 기술인 WebGL 과 HTML5 를 검토한 결과, 완전히 새로운 솔루션인 픽셀 스트리밍 플러그인을 개발하기로 했습니다.

이 문서에서는 콘텐츠 배포에 WebGL 과 HTML5 를 사용할 경우를 검토합니다. 특히 위에서 설명한 요구 사항 충족 여부에 관련된 분석과 검토한 내용을 중점적으로 다루며, 픽셀 스트리밍 기술을 개발한 이유도 함께 설명합니다.

WebGL

WebGL 은 상호작용형 2D 와 3D 그래픽을 렌더하고 사용자가 애플리케이션 또는 플러그인을 다운로드하지 않아도 그 결과를 웹 브라우저에서 표시하는 JavaScript API 입니다. WebGL 은 무료 오픈 스탠다드를 제작하는 비영리 조직 [크로노스 그룹\(Khronos Group\)](#)에서 개발 및 유지보수합니다.

WebGL 은 스마트폰, 태블릿, 비디오 게임 콘솔, PDA 와 같은 내장형 시스템을 위해 설계된 [OpenGL ES](#)(OpenGL for Embedded Systems)를 기반으로 제작되었습니다. WebGL 은 웹페이지에 그래픽을 그리는 HTML5 Canvas 엘리먼트를 통해 실행됩니다.

WebGL 은 피직스, 이미지 프로세싱, 이펙트의 GPU 가속 효과를 지원합니다.

WebGL 디플로이먼트는 두 가지 항목으로 구성됩니다.

- JavaScript 로 작성한 제어 코드
- OpenGL ES SL(OpenGL ES Shading Language)로 작성한 셰이더 코드

WebGL 은 정의상 디플로이먼트 전에 컴파일할 필요가 없습니다.

WebGL 저작도구

WebGL 로 콘텐츠를 구축하는 가장 효율적인 방법은 최종 결과를 보여주는 저작 소프트웨어를 사용하는 것입니다. 일반적으로 브라우저를 통해 온라인 테스트하는 툴을 이용하는 것을 의미합니다.

PlayCanvas - PlayCanvas Editor 는 고급 WebGL 저작 환경이며, 기본적으로 WebGL 게임 엔진입니다. JavaScript 로 2D 또는 3D 그래픽 시뮬레이션을 프로그래밍합니다. 모든 코드는 모든 주요 브라우저와 디바이스를 위한 표준을 따르는 크로스 플랫폼 HTML5 로 작성되었습니다.

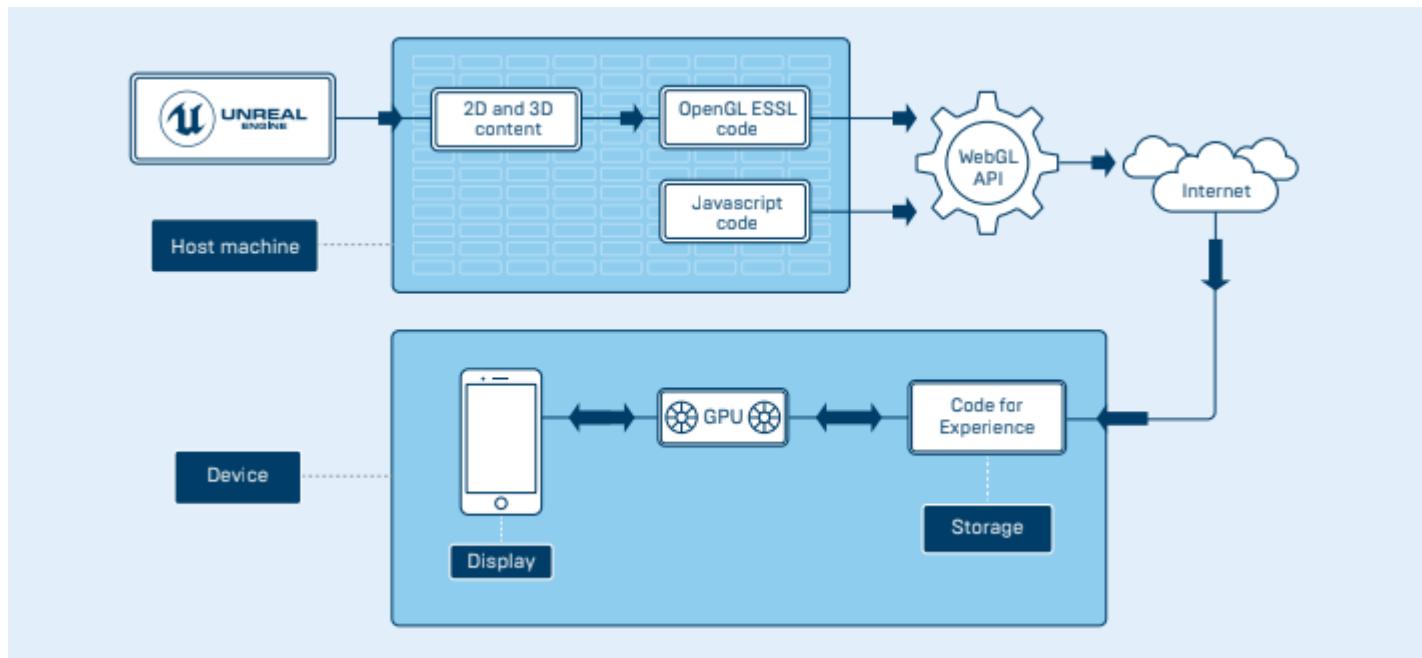
Sketchfab - Sketchfab 은 3D 콘텐츠를 공유하고 배포하는 온라인 마켓입니다. 3D 뷰어는 플러그인 없는 브라우저 배포 기능을 제공하며 VR 과 AR 을 지원합니다. 모든 콘텐츠는 Sketchfab 서버에서 호스팅하고 배포해야 합니다.

WebGL 제약사항

범위가 넓은 배포 툴로 사용할 경우 WebGL 의 주요 제약 사항은 콘텐츠 생성과 전달과 관련이 있습니다. 현재는 애플리케이션의 상호작용형 웹 디플로이먼트에만 사용 가능한 OpenGL ES SL 셰이더 언어로 실행하도록 콘텐츠를 제작해야 합니다. 언리얼 엔진과 같은 실시간 엔진으로 제작한 콘텐츠를 디플로이하려면 콘텐츠와 상호작용형 엘리먼트가 WebGL 프레임워크 내에서 실행되도록 구성해야 합니다.

또한 WebGL 은 클라이언트 측의 브라우저 수신 성능과 하드웨어에 완전히 의존합니다. 그래픽 퀄리티가 브라우저의 디스플레이 성능에 따라 정해집니다. 반드시 클라이언트 측에서 데이터를 다운로드해야 하고, 이 다운로드 시간이 경험 시작까지의 대기 시간을 좌우합니다. 대규모 데이터 세트가 필요한 복잡한 경험을 위해 클라이언트는 데이터 저장을 위한 공간을 확보해야 합니다.

민감하거나 기밀인 데이터인 경우 클라이언트 측의 데이터 보안을 위한 작업을 추가로 진행해야 합니다.



WebGL 콘텐츠가 다른 방식으로 제작된 콘텐츠와 혼합된 환경에서는 WebGL 퀄리티가 사용자 경험을 결정하는 기준이 됩니다. 더 넓은 플랫폼 관점에서 보면, 다른 엔드 채널과 연동할 때는 시스템이 최소 두 개의 데이터 세트, 셰이더 모델, 크리에이티브 구성을 유지해야 합니다.

메트릭스 측면에서는 웹사이트에서 사용한 일반 채널을 통해 로그인과 세션 시간 데이터를 수집합니다.

이런 제약에도 불구하고, WebGL은 작은 데이터 세트, 낮은 보안 민감도, 기본 퀄리티를 요구하는 경험에 적합한 아주 간단한 디플로이먼트 옵션입니다. 이러한 유형의 경험을 지원하기 위해서 에픽게임즈는 [glTF](#) 파일 형식과 셰이더 교환 지원 라이브러리를 포함한 WebGL을 UE4로 직접 익스포트하는 언리얼 엔진 옵션을 개발하고 있습니다. 2020년 1분기에 익스포터가 출시될 예정입니다.

에픽게임즈는 언리얼 엔진 경험에 대한 더 높은 수요에 부합하기 위해 더 적합한 솔루션을 계속 연구하고 개발해 왔습니다.

HTML5

HTML5는 HTML의 가장 최신 버전으로 웹페이지 개발을 위한 마크업 언어입니다. WebGL을 전혀 사용하지 않고 Canvas 엘리먼트로 HTML5 자체에 3D 경험을 개발하는 솔루션은 다양합니다. 이러한 솔루션을 사용하려면 GPU 프로그래밍 기술 지식이 필요하고 엔드

클라이언트가 다운로드할 폭넓은 데이터를 패키징해야 하기 때문에 사용자 경험의 퀄리티와 기능이 제한됩니다.

HTML5 제약 사항

이 방식은 애플리케이션을 클라이언트 하드웨어에서 실행하므로 사용자 경험이 시작되기 전에 클라이언트가 전체 데이터 세트를 다운로드해야 합니다. 이러한 디플로이먼트는 다운로드 속도와 저장 공간이 핵심 요소입니다. 풍부한 사용자 경험을 제공하려면 클라이언트에 기가바이트 단위의 데이터를 다운로드해야 합니다. 하지만 특히 모바일 디바이스에서는 대기 시간과 저장 공간 제한으로 인해 모든 데이터를 다운로드하지 못할 수 있습니다.

이 방법을 사용하려면 WebGL 과 같이 시스템이 두 가지 데이터 표준을 유지해야 합니다. 또한 원본 콘텐츠와 배포된 콘텐츠가 일관된 사용자 경험을 보장할 수 없습니다. 배포된 콘텐츠를 완전히 교체해야 업데이트와 유지가 가능하며, 이는 변동이 심한 콘텐츠에서 문제를 일으킬 수 있습니다. HTML5 에서도 WebGL 과 같은 보안 문제가 발생합니다.

언리얼 엔진은 프로젝트를 HTML5 플랫폼으로 퍼블리싱하는 툴을 제공합니다. 하지만 UE 4.24 버전부터 에픽은 HTML5 지원 기능을 커뮤니티 지원 플랫폼 확장 기능 형태로 GitHub 으로 이관했으며 공식 지원을 종료했습니다.

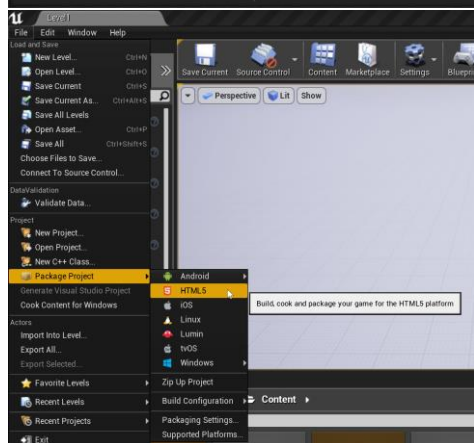
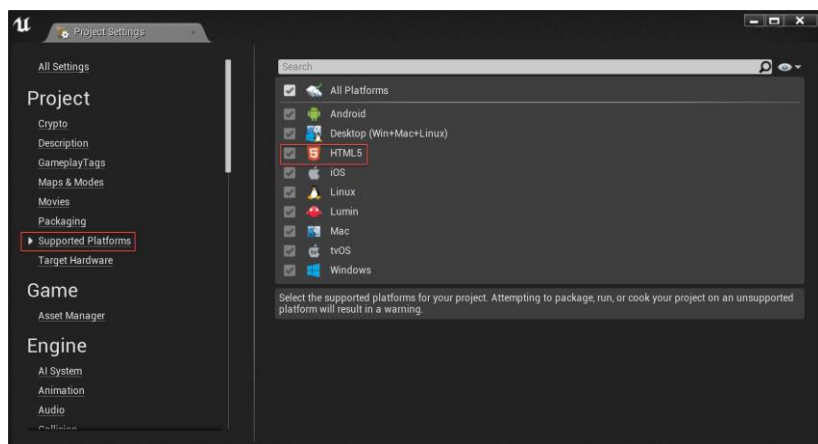


그림 x: 언리얼 엔진 인터페이스 안에서 HTML5 선택 화면

픽셀 스트리밍

에픽게임즈는 다양한 디바이스 유형에 실시간 콘텐츠를 배포하기 위한 최고의 솔루션을 찾는 과정에서 WebGL 과 HTML5 의 모든 톨과 기능을 검토했습니다. 이 기술의 제한 사항을 검토한 다음 픽셀 스트리밍이라는 기술을 직접 개발하기로 했습니다.

상호작용형 실시간 콘텐츠를 다양한 디바이스 유형에 배포하는 이상적인 솔루션에는 다음과 같은 기능이 포함되어야 합니다.

- 클라이언트 측에서 데이터를 다운로드하지 않고 호스트에서 픽셀을 스트리밍하는 기능
- 고성능 사용 사례와 저성능 사용 사례에서 동일한 단일 데이터 구성
- 플랫폼별 독립 디플로이먼트
- 모든 플랫폼과 디바이스에 대한 퀄리티 조정
- 높은 충실도와 퀄리티
- 모든 언리얼 엔진 기능을 공유하는 기능
- 직관적이고 단순한 유지보수와 업데이트
- 경험 시작 단계에서의 빠른 접속
- 배포 전, 중, 후의 데이터 및 구성에 대한 보안
- 여러 사용 사례를 지원하는 다양한 컨피규레이션
- 단일 서버 또는 클라우드에서 디플로이가 가능한 확장성
- 사용자와 세션의 메트릭스 추출 기능

픽셀 스트리밍은 언리얼 엔진 플러그인으로 구현되었습니다. 이 플러그인은 호스트 서버에서 그래픽 스트림을 인코딩해서 WebRTC 프로토콜을 통해 수신 브라우저와 디바이스로 전송합니다. 결과적으로 고성능 호스트 시스템에서 실행되는 언리얼 엔진 애플리케이션은 최종 사용자의 디바이스에서 호스트 디바이스와 같은 퀄리티로 모든 언리얼 엔진 기능을 활성화합니다.

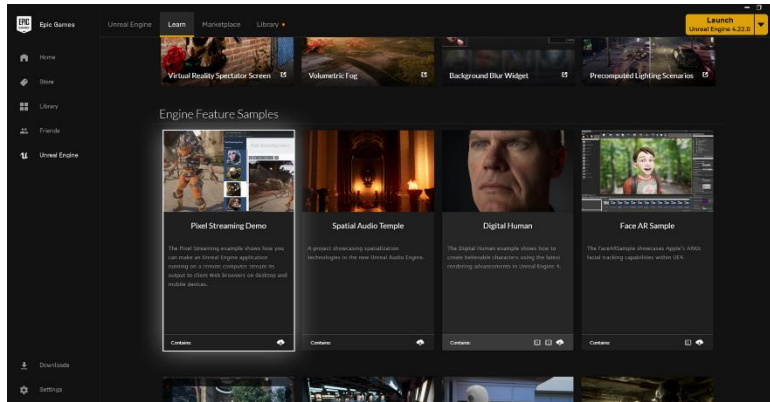


그림 3: 런처 인터페이스에서 본 픽셀 스트리밍 플러그인

데이터는 호스팅 서버에 남아 있고 픽셀만 사용자의 디바이스에 배포되기 때문에 픽셀 스트리밍과 같은 스트리밍 솔루션은 클라이언트에서 다운로드하는 솔루션보다 본질적으로 더 빠르고 더 안전합니다. 또한 UE4 안에서는 필요한 메트릭스에 따라 사용자 세션 데이터를 추출해 저장합니다

WebRTC 프로토콜

WebRTC(Web Real-Time Communication)는 웹 브라우저와 모바일 애플리케이션에서의 실시간 커뮤니케이션(Real-Time Communication, RTC)을 위한 프로토콜입니다. 이 프로토콜에서는 사용자가 플러그인 또는 앱을 다운로드하지 않아도 다이렉트 링크를 통해 오디오와 비디오를 전송합니다. 커뮤니케이션 명령어는 API(application programming interface)를 통해 제출합니다.

요구 사항

픽셀 스트리밍은 단일 서버에서 또는 적합한 하드웨어 수준으로 다이내믹 스케일링과 프로비저닝이 가능한 GPU 클라우드 환경에서 실행 가능합니다. 이런 상황에서의 핵심 요소는 필요에 따른 스케일 분석입니다. 최종 호스트 환경의 비용 및 사용자 경험 속도와 직접적인 관련이 있기 때문입니다.

픽셀 스트리밍 플러그인은 호스트 서버에서 WebRTC 를 통해 연결된 서버 또는 클라이언트와 커뮤니케이션합니다. 가장 단순한 형태는 로컬 IP 주소와 80, 8124, 8888 네트워크 포트로 접속하는 로컬 호스트입니다.

비디오 카드

NVIDIA GPU(Kepler 세대부터)에는 그래픽 성능으로부터 독립되고 하드웨어 기반 완전 가속 비디오 인코딩을 지원하는 하드웨어 기반 인코더 NVENC 가 있습니다. 그래픽 엔진과 CPU 는

전체 인코딩을 하는 복잡한 연산 과제를 NVENC 로 오프로드해 다른 작업에 대한 부담을 줄입니다. NVENC 의 효과는 다음과 같습니다.

- CPU 를 활용하지 않고 게임과 애플리케이션을 아주 낮은 지연시간과 높은 퀄리티로 인코딩하고 스트리밍합니다.
- 아카이빙, [OTT 스트리밍\(OTT streaming\)](#), 웹 비디오 서비스에서 아주 높은 퀄리티로 인코딩합니다.
- 스트림 당 초저전력(스트림 당 와트)으로 인코딩합니다.

비디오 인코딩 하드웨어인 NVENC 는 NVIDIA 의 비디오 코덱 SDK 를 이용해 접근합니다. 해당 전용 가속기는 Windows 와 Linux 에서 수요가 높은 다양한 비디오 코덱에 대한 하드웨어 가속 인코딩을 지원합니다.

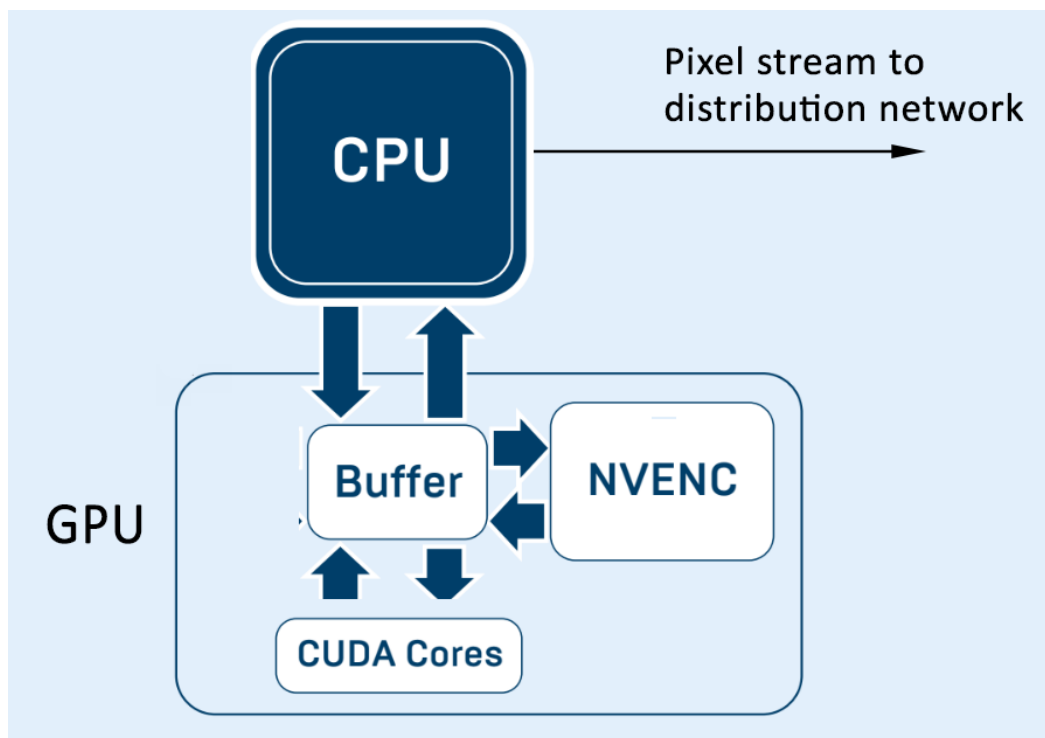


그림 4: 하드웨어 기반 인코딩과 픽셀 스트림 생성 NVENC 인코더는 버퍼를 통해 CPU 와 CUDA 코어와 연동해 픽셀 스트림을 생성하고, 이 픽셀 스트림은 CPU 를 통해 배포 네트워크로 전해집니다.

자세한 내용은 [NVIDIA 비디오 코덱 SDK 문서](#)를 참고하시기 바랍니다.

GPU의 표준이 개발되고 드라이버가 더 많이 보급되면서 언리얼 엔진은 4.24 버전부터 픽셀 스트리밍의 AMD 옵션을 지원합니다. 이후 출시 버전에도 더 많은 하드웨어 지원 옵션을 추가할 예정입니다.

픽셀 스트리밍 사용 사례

픽셀 스트리밍을 위한 흔한 구성과 이를 지원하는 기술을 함께 살펴보겠습니다. 자세한 내용은 언리얼 엔진 픽셀 스트리밍 문서에서 [호스팅과 네트워킹 가이드](#) 주제를 참고하시기 바랍니다.

호스트와 클라이언트가 네트워크 방화벽에 의해 분리된 경우, 호스트와 참가자 사이의 커뮤니케이션 구축을 위해 STUN(Session Traversal Utilities for NAT)과 TURN(Traversal Using Relays around NAT) 서버가 필요할 수 있습니다. 어떤 클라이언트 방화벽은 일반 제한 사항으로 호스트 서비스에서 또는 호스트 서비스로의 접속을 허용하지 않습니다. 이러한 상황에서 STUN/TURN 서버는 호스트 접속을 요청하는 클라이언트 사이의 커뮤니케이션 경로를 형성하기 위해 다양한 방법을 시도합니다. 또한, 이 방법으로 방화벽에 가려진 사용자의 문제를 해결할 수 있습니다. 사용자는 이 방법을 사용하지 않으면 스트리밍 앱을 이용할 때마다 시스템 관리자에게 특별 권한을 요청해야 합니다.

LAN 내 협업

가장 기본적인 공유 구성에서는 호스트와 개발 환경을 공유해야 하는 두 개 이상의 사용자 집단이 모두 같은 로컬 네트워크(LAN)를 사용합니다. 콘텐츠는 내부에 포함된 서버 플러그인을 통해 스트리밍되고, 호스트와 참가자 사이의 커뮤니케이션은 WebRTC API를 이용합니다.

LAN이 방화벽으로 구분된 상황이라면 STUN/TURN 서버가 필요할 수도 있다는 점을 유의하시기 바랍니다.

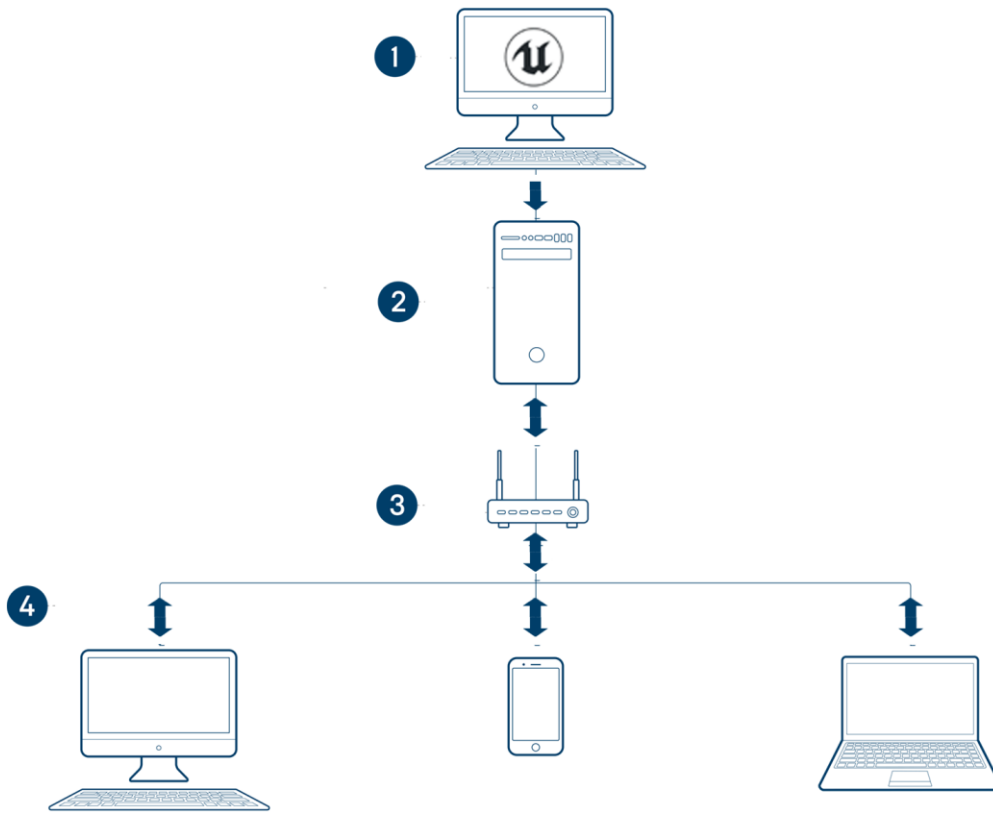


그림 5: LAN 내 단순 협업 상황

핵심: 1. 언리얼 엔진 개발 2. WebRTC 와 LAN 에서 UE4 서버 앱 3. 라우터 4. 사용자 디스플레이 상호작용

커뮤니케이션, 승인, 프레젠테이션

별도 LAN 에서 각각의 방화벽을 사용하는 특정하고 알려진 원격 참가자에게 발표하거나 프로젝트를 논의하기 위해 충실도가 높은 시스템이 필요한 경우에 픽셀 스트리밍은 제품 데이터를 전송할 필요가 없는 안전한 커뮤니케이션 방법을 제공합니다. 이런 구성에서는 호스트와 클라이언트 사이의 커뮤니케이션을 관리하기 위해 STUN/TURN 서버가 필요합니다.

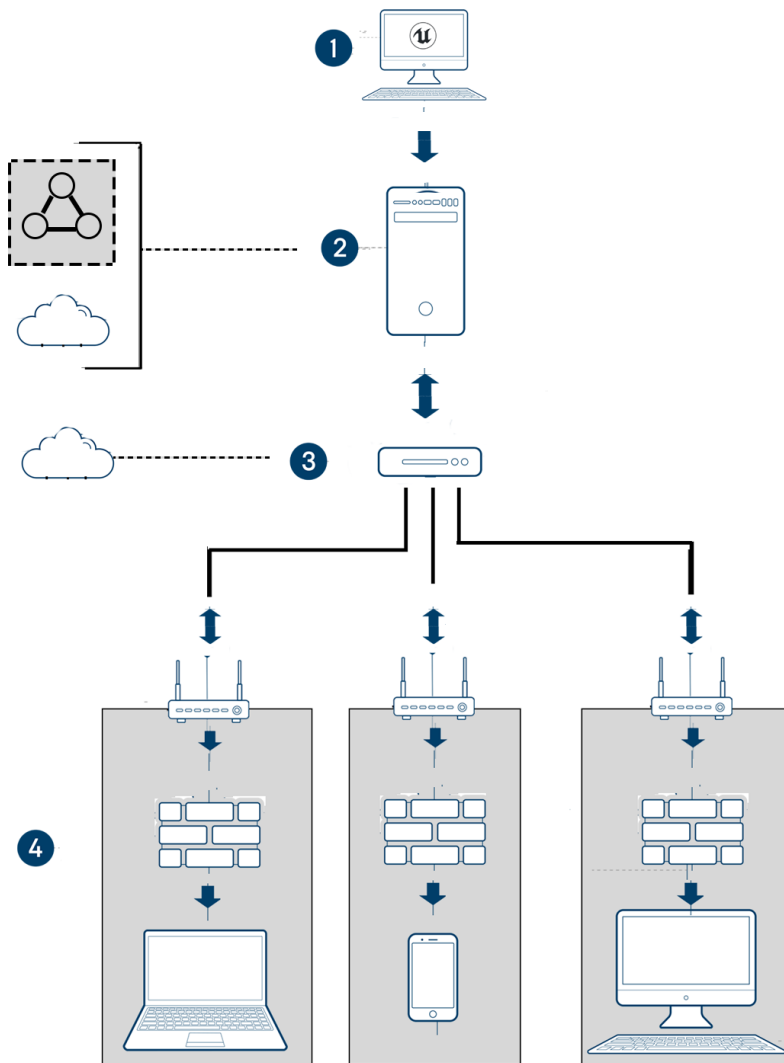


그림 6: 온프레미스 네트워크 또는 클라우드에서의 단일 인터페이스 연결

핵심: 1. UE4 개발 2. 온프레미스 네트워크 또는 클라우드의 UE4 서버 앱 3. 클라우드의 STUN/TURN 서버 4. 라우터/ 클라이언트 네트워크 진입점 5. 방화벽 6. 디스플레이 상호작용 디바이스

원격 협업 디자인, 공유 경험, 소비자 애플리케이션, 컨피규레이터

다양한 하드웨어와 소프트웨어 구성을 보유한 많은 최종 사용자에게 애플리케이션의 서비스를 제공해야 한다면, 확장 가능한 클라우드 환경에서 픽셀 스트리밍을 실행할 수 있습니다. 이 구성은 자동차 컨피규레이터 또는 안전한 데이터 접속이나 추적이 가능한 경험이 필요한 다른 공유 경험에 적합한 전형적인 구성입니다. 이러한 구성은 여러 개발팀이 원격으로 협업하여 제품을 디자인하는 상황에도 적합합니다.

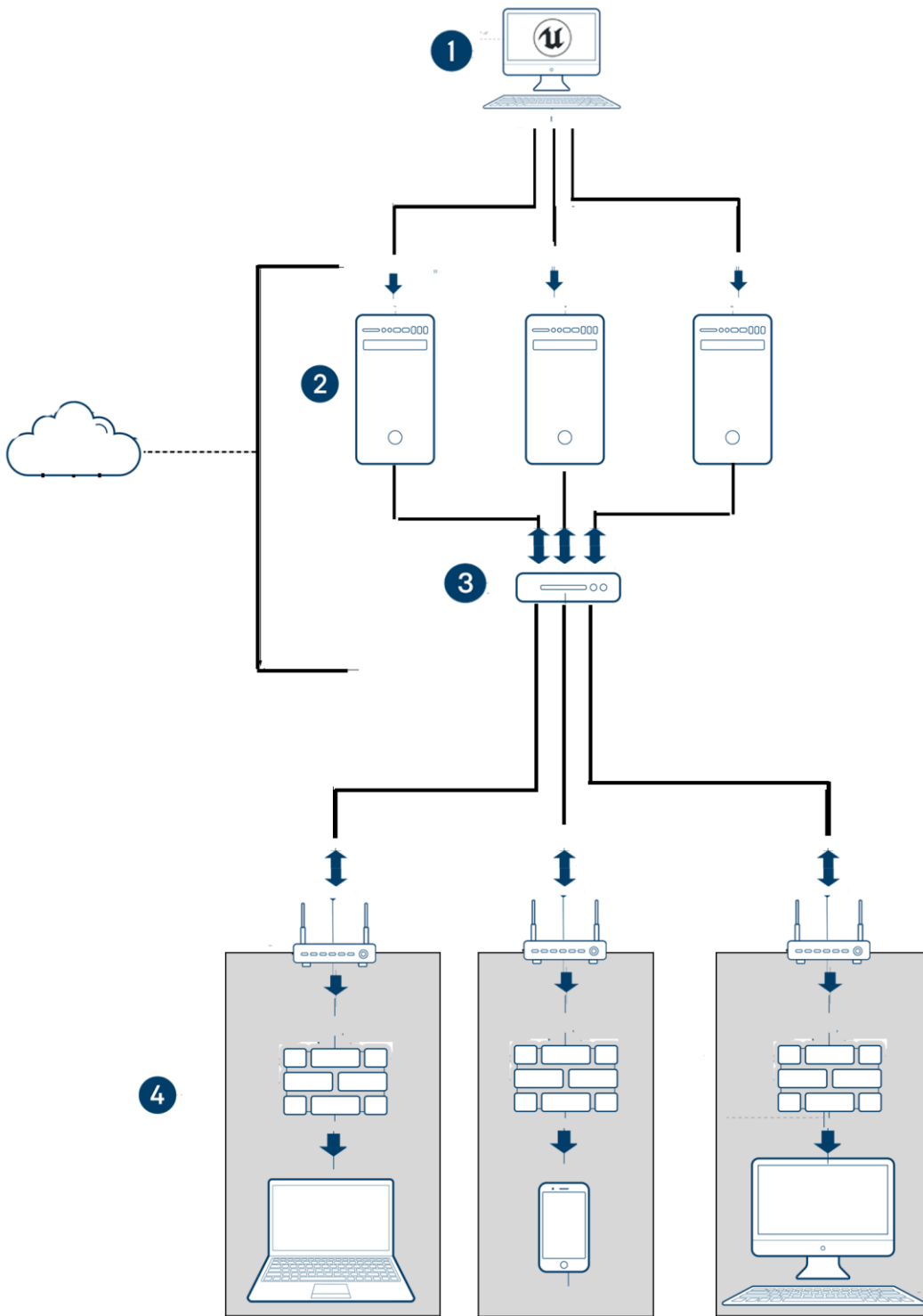


그림 7: 모든 사용자를 위한 독특한 단일 경험

핵심: 1. 언리얼 엔진 개발 2. 클라우드의 Web RTC 와 UE4 서버 앱 3. STUN/TURN 서버 4. 사용자 디스플레이 상호작용

대부분의 상용 사례에서 각 사용자는 각자의 상호작용 경험과 스트림이 필요합니다. 시스템은 사용자별 픽셀 스트리밍 컴포넌트의 개별 스택을 실행하면서 각 사용자를 별도의 웹서버나 해당되는 경우에는 호스트로 안내합니다.

각 스택에는 사용자 경험을 제어하기 위한 개별 포트와 식별자가 필요합니다. 많은 소비자 수준의 그래픽 카드는 최대 두 개의 인코더만 동시 실행할 수 있으며, 이로 인해 컴퓨터에서 실행하는 인스턴스 수가 제한됩니다. NVIDIA Quadro 와 Tesla 와 같은 전문가 수준의 그래픽 카드 또는 클라우드 기반 GPU 인스턴스(AWS)에는 이런 제한이 없습니다.

매치메이킹 시스템이 각 요청자를 클라이언트와 그 WebRTC 프록시 서버 사이의 연결을 생성하는 대상 웹서버로 리디렉트해 줍니다. 사용자가 서버에서 활동하고 있다면 해당 매치메이킹 시스템은 해당 사용자에게 대한 스트림을 유지해 줍니다. 매치메이킹 시스템 컴포넌트는 다른 서버 컴포넌트와 마찬가지로 언리얼 엔진 안에서 찾을 수 있습니다. 자세한 정보는 언리얼 엔진 픽셀 스트리밍 문서에서 [호스팅과 네트워킹 가이드](#) 주제 내의 *매치메이킹과 다중 전체 스택* 항목을 참고하시기 바랍니다.

요약

에픽게임즈에서 고퀄리티의 언리얼 엔진 콘텐츠를 다양한 디바이스 유형에 스트리밍하는 솔루션들을 검토한 결과, 픽셀 스트리밍을 개발하기로 했습니다.

WebGL 은 현재 호스트 환경 내에서 디플로이할 때 비용이 부담스럽지 않으며 간단한 사용 사례에서 충분히 유효한 솔루션입니다. 에픽게임즈는 언리얼 엔진을 위한 WebGL 익스포터를 개발하고 있지만 간단한 구현에만 적합한 익스포터입니다.

HTML5 솔루션은 이미 최대한 활용할 수 있는 한계에 있으며 이 영역에서는 상용을 목표로 개발되고 있지 않습니다. 이런 솔루션들의 디플로이먼트는 특정한 사용사례와 애플리케이션에 좁혀진 맞춤형 개발에 의존하게 될 것입니다.

픽셀 스트리밍은 최고의 콘텐츠를 가능한 한 가장 높은 충실도로 배포하는 일이 핵심인 경험의 시대에, 콘텐츠를 배포하기 위한 사용자 및 기술 관련 고려사항을 모두 충족합니다. 픽셀 스트리밍은 최종 사용자나 플랫폼을 고려하지 않고도 상위 경험을 공유할 수 있으며, 더 큰 플랫폼 구성에서도 잘 실행되며 오프라인과 온라인 콘텐츠를 위한 다양한 채널도 제공합니다.

심화 단계

UE4 안에서 픽셀 스트리밍을 구성하는 방법에 대한 자세한 내용은 아래 링크를 참고하시기 바랍니다.

<https://launcher-website-prod07.ol.epicgames.com/ue/learn/pixel-streaming-demo>

단계별 설명은 다음을 참고하시기 바랍니다.

<https://docs.unrealengine.com/ko/Resources/Showcases/PixelStreamingShowcase/index.html>

본 문서 소개

저자

하이코 웬젤(Heiko Wenczel)

기여자

세바스찬 미글리오(Sébastien Miglio)

마르코 아나스타시(Marco Anastasi)

편집

미셸 부스케(Michele Bousquet)