



“팀의 프로젝트 성공을 위해
준비되어 있는 개발자입니다.”

유영재

경력 : 4년 6개월

나이 : 36세

연락처

- 이메일 : gameproyyj2@gmail.com
- 전화 : 010-7130-0637
- Git : github.com/yooyoungjae0627

학력

- 한국 공학 대학교 게임 공학과 (편입)
2017년 3월 ~ 2020년 8월 (학사 학위)
- 명지 전문 대학 영어과
2013년 3월 ~ 2017년 2월
- 15사단 39연대 2대대 GOP 경계 헌병
2014년 4월 ~ 2015년 12월 (만기 제대)

경력 사항

(주) 모비릭스

MMORPG 신규 제작팀

기간: 2024.03 ~ 2025.12 (1년 10개월)

직무: MMORPG UE5 클라이언트 프로그래머

주요 업무

- UE5 기반 MMORPG 신규 제작팀에서 주요 콘텐츠 제작을 담당하며 클라이언트 C++ 개발 수행
- 제작, 거래소, 룬, 필드맵, 인벤토리 등 MMO 핵심 콘텐츠 시스템을 신규 개발 단계에서 구현
- 신규 콘텐츠 제작을 위해 공용 UI 구조를 설계·구현하고, 다양한 콘텐츠에 재사용 가능하도록 확장성 확보
- 팀 공통 Req → Ack 기반 클라이언트-서버 통신 구조에 맞춰 콘텐츠 로직 및 UI 연동 구현
- 기획·서버·아트와 협업하며 신규 콘텐츠 개발, 기능 보완, QA 피드백 대응을 지속 수행

(주) 액션스퀘어

삼국 블레이드 라이브 서비스팀

기간: 2021.07 ~ 2024.02 (2년 8개월)

직무: Mobile 액션 RPG UE4 콘텐츠 프로그래머

주요 업무

- 라이브 서비스 빌드의 안정성 유지 및 기능 개선
- 신규 이벤트/콘텐츠 UI 및 로직 구현
- 기획/아트/서버와의 협업을 통한 대규모 업데이트 대응

What I Did at 모비릭스

(주) 모비릭스

MMORPG 신규 제작팀 주요 구현 콘텐츠

MMO 공용 아이템 인벤토리 UI , 공용 아이템 아이콘 UI 프레임워크

- [공용 아이템 인벤토리, 아이템 아이콘 프레임워크 시연 영상 링크 \(클릭\)](#)

- 게임 전역(인벤토리·장비·보상·제작·거래소·창고·퀵슬롯)에서 재사용 가능한 공통 인벤토리 UI 프레임워크 설계
- 카테고리 탭·아이콘 그리드·상세 패널 구조의 통합 인벤토리 UI 시스템 구현
- RedDot·검색·필터·즐거찾기·분해 필터를 단일 시스템으로 통합
- 공통 아이콘 위젯과 마커 정책(장착·잠금·기간·스택 등) 일관화로 UI 유지보수성과 확장성 확보

MMO 공용 아이템 상세 정보 페이지 프레임워크

- [공용 아이템 상세 정보 페이지 프레임워크 시연 영상 링크 \(클릭\)](#)

- 아이템·스킬·룬·버프 등의 상세 정보를 단일 공용 Detail UI 프레임워크로 통합
- DetailSub 데이터 기반 레이아웃 구성과 단일 컨텍스트 구조체로 데이터 드리븐 UI 설계
- 동적 UI 조립 + 필드 UI 풀링을 통해 반복 진입 시에도 성능 안정성 확보
- 강화·거래소·제작·컬렉션 등 TID 기반 콘텐츠 연동 및 장비 비교 기능 확장

MMO 룬 시스템 (Client / UI Architecture)

- [룬 시스템 시연 영상 링크 \(클릭\)](#)

- 룬 인벤토리, 결합력, 합성 기능을 단일 UI 프레임워크로 통합 구현
- 인벤토리 / 결합력 / 합성 모드를 전환하는 구조와 슬롯 상태머신 기반 입력 처리 설계
- 합성 결과 연출 이후 인벤토리·RedDot·UI가 자동 갱신되도록 전체 흐름 구성
- 장착/해제/합성 패킷 연동 및 등급별 확률·비용 테이블 기반 UI/로직 처리

What I Did at 모비릭스

(주) 모비릭스

MMORPG 신규 제작팀 주요 구현 콘텐츠

MMO 필드맵 시스템 (Client / UI Architecture)

- [필드맵 시스템 시연 영상 링크 \(클릭\)](#)

- 뷰포트는 고정하고 지도 요소를 MapPanel 단일 컨테이너로 관리하여 패닝·줌 시 UI 정합성과 유지보수성 확보
- 지형 터치 기반 NavPath 자동 이동, 이동 경로 표시·이펙트·취소 UI까지 통합 구현
- 월드 좌표 → 지도 UI 좌표 변환 구조로 플레이어·지역·포탈 아이콘을 줌/이동과 무관하게 정확히 동기화
- 기억 장소 저장, 순간 이동, 지역 즐겨찾기 등 탐색 편의 기능 확장 구조 설계
- 지도 여백 제한, 이동 범위 자동 보정, 해상도 대응 UX 재설계를 통해 안정적인 필드맵 인터랙션 완성

MMO 거래소 시스템 (Client / UI Architecture)

- [거래소 시스템 시연 영상 링크 \(클릭\)](#)

- 구매, 판매, 개인거래, 정산, 거래내역을 단일 거래소 UI로 통합 설계
- 다중 거래 카테고리 State 전환 기반 UI 구조로 관리
- UI 입력 → 서버 Req → Ack → 데이터 갱신 → UI 자동 리프레시 흐름 구현
- Noti 패킷 기반 레드닷 UI로 실시간 거래 이벤트 인지 구조 구현
- 다단계 리스트 및 다수 거래 패킷을 안정적으로 처리하는 MMO 거래소 시스템 구현

MMO 제작 시스템 (Client / UI Architecture)

- [제작 시스템 시연 영상 링크 \(클릭\)](#)

- 아이템 선택, 재료 구성, 수량 설정부터 서버 요청 및 결과 처리까지 전체 제작 흐름을 구성
- Req → Ack 기반 서버 연동을 통해 제작 결과에 따른 데이터 동기화 및 UI 갱신 처리
- 성공/대성공/실패 분기를 갖는 결과 팝업 UI 흐름을 구조화하여 구현

What I Did at 액션스퀘어

(주) 액션스퀘어

삼국 블레이드 라이브 서비스팀 대표적인 주요 구현 콘텐츠

징표 교환소 시스템

- [징표 교환소 시스템 시연 영상 링크 \(클릭\)](#)

- 징표 재화 기반 교환 아이템 구성
- 교환 가능 여부 판단 및 UI 상태 제어
- 서버 Req → Ack 응답에 따른 교환 결과 처리
- 보상 지급 이후 인벤토리 및 UI 자동 갱신

코스튬 시스템

- [코스튬 시스템 시연 영상 링크 \(클릭\)](#)

- 코스튬 장착 / 해제 / 미리보기
- 외형 변경 및 능력치 반영
- 서버 연동 데이터 기반 상태 갱신
- UI 입력 → 서버 요청 → 결과 반영 → UI 자동 갱신 파이프라인

회사 홍보 영상 링크

[여름 대규모 업데이트 수영복 코스튬 홍보 영상 링크 \(클릭\)](#)

What kind of programmer are you?

개발자 유영재는

구조와 책임을 먼저 설계하는 개발자입니다.

라이브 서비스와 신작 초기 개발을 모두 경험하며, 저는 기능 구현 자체보다 구조와 책임 분리가 프로젝트의 안정성과 개발 속도를 좌우한다는 점을 실무에서 체감하며 성장해왔습니다.

특히 라이브 서비스 팀에서 일하면서 구조와 기준의 중요성을 가장 크게 느꼈습니다. 당시 팀에는 비교적 많은 신입 개발자들이 투입되어 있었고, 라이브 환경 특성상 빠른 대응과 기능 추가가 반복되는 상황이었습니다. 여러 개발자가 동시에 수정과 개선을 진행하다 보니, 코드와 로직이 기능 단위로 확장되면서 구조적 일관성을 유지하기 어려운 지점들이 생기기 시작했습니다.

그 결과 기능 하나를 수정했을 뿐인데 전혀 연관 없어 보이는 다른 화면이나 시스템에서 문제가 함께 발생하는 상황을 여러 차례 경험했습니다.

당시 프로젝트는 Req → Ack 기반 요청/응답 구조에 더해, 서버에서 상태 변화를 Push하는 Noti 패킷이 함께 동작하는 구조였습니다. 즉, 클라이언트 상태는 서버 응답과 비동기 Noti를 통해 최종적으로 확정되는 방식이었습니다.

그러나 클라이언트 내부에서는 서버 Ack 처리, Noti 처리, UI 이벤트, 화면 전환 로직이 동일한 상태 값을 각각 직접 수정하는 구조가 일부 존재했습니다. 그 결과 서버에서 확정된 값과 로컬에서 임의로 변경된 값이 엇갈리는 상황이 발생했고, 어떤 값이 최종 상태인지 추적하기 어려워졌습니다.

특정 타이밍에서만 발생하는 버그가 반복되었고, QA 단계에서 예상치 못한 수정이 누적되며 운영 비용 역시 증가했습니다. 이 경험을 통해 저는 깨달았습니다. 기능을 빠르게 구현하는 것보다, 상태가 어디에서 확정되고 누가 변경 책임을 갖는지를 먼저 정의하는 구조가 훨씬 중요하다는 것을.

이후 저는 기능을 구현하기 전에 먼저 기준을 정리하는 방식을 택했습니다.

- 권한은 서버 응답(Ack)과 Noti 데이터에만 둔다.
- 상태를 변경하는 책임은 상위 매니저 계층으로 한정한다.
- 하위 UI는 상태를 직접 수정하지 않고, 가공된 결과를 표현만 한다.

기능을 추가할 때마다 해당 로직이 공통 구조로 사용되어야 하는지, 아니면 특정 영역이 독립적으로 책임져야 하는지를 먼저 구분하고, 상태의 결정 지점과 권한의 위치를 명확히 정의한 뒤 개발을 진행했습니다. 초기 구현 속도는 다소 느릴 수 있었지만, 기능이 누적될수록 변경 범위를 빠르게 예측할 수 있었고 테스트 부담도 크게 줄어들었습니다.

What kind of programmer are you?

개발자 유형재는

구조와 책임을 먼저 설계하는 개발자입니다.

이러한 접근은 공용 프레임워크 설계로 이어졌습니다. 인벤토리, 보상, 제작, 거래소 등 여러 화면에서 반복 사용되는 아이콘 및 인벤토리 UI를 개별 구현하지 않고, 공통 데이터 구조와 단일 갱신 흐름을 갖는 프레임워크 형태로 통합했습니다. 아이콘 위젯을 단일 컴포넌트로 구성하고, 장착·잠금·기간·스택 등의 마커 정책을 일관화했으며, RedDot·검색·필터 구조 역시 통합했습니다. 그 결과 신규 콘텐츠가 추가되더라도 정책과 데이터 확장만으로 대응할 수 있었습니다.

상세정보 UI 역시 대상별로 나누지 않고 단일 컨텍스트 기반 구조로 통합했습니다. 특히 반복적으로 열리고 닫히는 특성을 고려해, 위젯을 매번 생성·파괴하지 않고 재사용하는 구조로 설계했습니다. 이는 Object Pooling 개념을 UI 레벨에 적용한 것으로, 반복 진입 시 생성 비용과 GC 부담을 줄이고 성능 안정성을 확보하기 위한 선택이었습니다.

이러한 실무 경험을 바탕으로 포트폴리오에서는 서버 구조 자체를 직접 설계했습니다. Unreal Engine Dedicated Server 환경을 전제로, 권한·상태·책임을 처음부터 명확히 분리하는 것을 핵심 기준으로 삼았습니다.

권한은 Dedicated Server에만 두고, 게임 결과에 영향을 주는 모든 판단과 상태 변경은 서버에서만 이루어지도록 설계했습니다. 개인 상태는 PlayerState, 전역 상태는 GameState에만 두어 어디를 보면 현재 게임 상태를 알 수 있는지가 명확하도록 구성했습니다. UI는 상태를 직접 수정하지 않고, 서버에서 확정된 값이 Replication과 RepNotify를 통해 전달될 때만 갱신하도록 분리했습니다.

전투 시스템 역시 같은 원칙으로 설계했습니다. Gameplay Ability System을 기반으로 Ability → GameplayEffect → AttributeSet의 단일 파이프라인을 구성했고, 클라이언트는 입력만 전달하며 실제 판정과 데미지 계산은 서버에서 수행합니다. 서버가 트레이스를 통해 히트 대상과 위치를 확정하고 최종 DamageValue를 계산한 뒤 SetByCaller 방식으로 전달하며, 체력 감소와 사망 확정은 AttributeSet에서만 처리하도록 분리했습니다. 이를 통해 계산과 규칙을 명확히 구분하고 수정 지점을 단일화했습니다.

저는 단기적으로 빠르게 동작하는 기능보다, 장기적으로 유지 가능하고 확장 가능한 구조를 먼저 설계하는 개발자입니다. 실무에서 다듬은 구조적 기준을 바탕으로, 포트폴리오에서는 그 원칙을 서버 권위 기반 멀티플레이 아키텍처까지 확장해 구현했습니다.

What kind of programmer are you?

개발자 유영재는

팀워크를 바탕으로 프로젝트 성공에 기여하는 개발자입니다.

실무에서는 일정과 기획 변경이 발생하는 상황을 자주 경험했습니다. 기획 회의를 통해 방향과 데드라인이 정해진 이후에도, 구현 도중 기능이 추가되거나 UI 리소스가 늘어나는 경우가 있었습니다. 이러한 상황에서 저는 기능을 다음 빌드로 미루기보다, 가능한 범위 내에서는 이번 빌드에서 완성도를 맞추는 방향을 선택해왔습니다. 중간 단계로 넘길 경우 QA 비용이 증가하고 이후 수정 범위가 확장될 가능성이 높다고 판단했기 때문입니다.

다만 기획 방향이 프로젝트 운영이나 유저 경험 측면에서 부담이 될 수 있다고 판단될 때는, 우선 예상 가능한 영향과 리스크를 정리하는 데 집중했습니다. 특히 결정 권한을 가진 담당자가 있는 경우에는 시니어 기획자나 시니어 프로그래머와 먼저 검토하며 리스크를 공유하고, 근거를 정리한 뒤 방향을 조율했습니다. 개인적인 의견 제시가 아니라, 데이터와 구조적 영향 분석을 바탕으로 판단 기준을 명확히 전달하는 방식을 택했습니다.

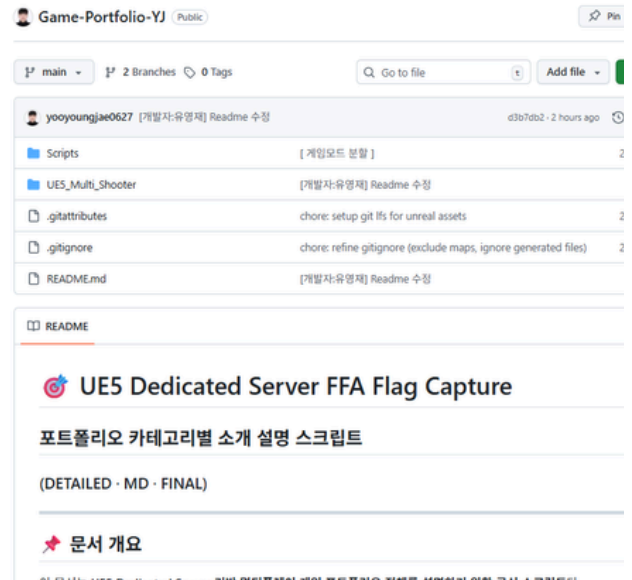
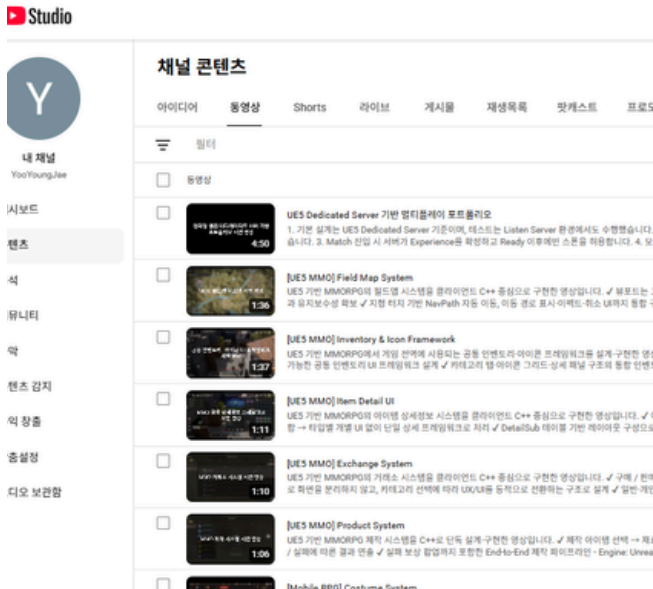
대표적인 사례가 룬 시스템이었습니다. 룬 장착 해제 시 확률적으로 룬이 파괴되는 기획이 있었는데, 시장 트렌드와 유저 경험을 고려했을 때 장기적인 이탈 가능성이 있다고 판단했습니다. BM 트렌드, 유저 반응 패턴, 라이브 이후 CS 리스크를 근거로 정리해 공유했고, 결과적으로 유저 부담을 완화하는 방향으로 개선되었습니다. 저는 이러한 과정에서 항상 근거와 구조를 중심으로, 팀 전체가 납득할 수 있는 방향을 찾는 것을 중요하게 생각합니다.

이러한 경험을 바탕으로, 지금까지 수행해온 UE5 클라이언트 개발 경험과 구조를 기준으로 시스템 완성도를 높여온 방식이 회사에서 요구하는 역할과 잘 맞는다고 판단했습니다. 기존 경험을 효과적으로 활용하면서, 새로운 환경에서 더 넓은 책임과 역할을 수행하며 역량을 확장하고 싶습니다.

단기적으로는 UE5 클라이언트 역량을 더욱 깊게 다지고, 팀 내에서 신뢰받는 개발자로 자리 잡는 것이 목표입니다. 장기적으로는 AI 시대에도 유연하게 대응할 수 있도록 기술적 폭을 확장해, 혼자서도 하나의 게임을 완성할 수 있는 역량을 갖춘 개발자로 성장하고자 합니다.

변화하는 환경 속에서도 기준을 세우고 구조를 설계하며, 프로젝트의 완성도를 높여 팀과 함께 지속적인 성과를 만들어가는 개발자가 되겠습니다.

컨텐츠 시연 영상 링크 모음



MMO 공용 아이템 인벤토리 UI , 공용 아이템 아이콘 UI 프레임워크
<https://www.youtube.com/watch?v=ERBLx1N5tmc>

MMO 공용 아이템 상세 정보 페이지 프레임워크
<https://www.youtube.com/watch?v=uivB-4JgvKA>

MMO 필드맵 시스템
<https://www.youtube.com/watch?v=gG3ip175oWQ&feature=youtu.be>

MMO 룬 시스템
<https://www.youtube.com/watch?v=ciTpvWkPSR8>

MMO 거래소 시스템
<https://www.youtube.com/watch?v=WNKcZvpgVVE&feature=youtu.be>

MMO 제작 시스템
<https://www.youtube.com/watch?v=hEYmYV3ffmw>

액션 RPG 징표 교환소 시스템
<https://www.youtube.com/watch?v=MNlL3jS8Aoo>

액션 RPG 코스튬 시스템
<https://www.youtube.com/watch?v=HFNNvTjhLU8&feature=youtu.be>

포트폴리오 GIT HUB(깃허브) 링크
<https://github.com/yooyoungjae0627/Game-Portfolio-YJ>

포트폴리오 유튜브 영상 링크
<https://www.youtube.com/watch?v=JLWoK7s5ayU>