# 캡스톤디자인

## **3주차.** 아키텍쳐 설계 및 개발 환경 구성

2025년 1학기
교수 김하연

# 왜 아키텍처가 중요할까?

- 아키텍처는 단순한 설계도가 아닌 비즈니스 리스크 관리의 핵심
- **4학년이라면 이제는 확장성, 유지보수성, 협업 효율을 고려해야 함 ^^b**
- 기술 부채의 실제 비용
  - 리팩토링에 6개월 소요
  - 시장 점유율 상실
- 현재 트렌드
  - 컨테이너/Kubernetes
  - DevOps/MLOps
  - 클라우드 네이티브 아키텍처

# 웹/앱 아키텍처

- MVC, MVVM, MVP 패턴 선택은 팀 구조와 비즈니스 요구사항에 맞춰야 함
- 서버리스 아키텍처의 제약
  - 콜드 스타트, 벤더 종속성, 디버깅 어려움
- API 디자인 선택의 영향
  - REST vs GraphQL
  - 동기 vs 비동기
- 아키텍처 선택은 비즈니스 전략, 팀 구성, 확장 계획을 반영하는 의사결정

# 웹/앱 아키텍처 팁..?

- 초기 스타트업 구조
  - React + Node.js/Express + MongoDB
- 주요 병목: 데이터베이스 확장성, 모노리틱 백엔드 한계
- 백엔드 분할 전략: 수직적(기능별) vs 수평적(데이터별) 분할
- 캐싱 전략의 중요성: Redis로 DB 부하 90% 감소 가능
- 단계별 아키텍처 진화: 초기(~10만 사용자) → 성장(~100만) → 확장(~1000만)
- 점진적 진화의 중요성
  - **"빅뱅" 방식 전환은 대부분 실패**

# 게임 아키텍처

- 게임 루프의 중요성: Input → Update → Render
- 사이클 최적화 사례: 시간 슬라이싱으로 NPC 업데이트 성능 4배 향상
- 싱글플레이 vs 멀티플레이 아키텍처 차이
- 멀티플레이 핵심 요소: 권한 모델, 상태 동기화 전략, 레플리케이션 최적화
- 플레이어 경험 중심의 아키텍처 결정 중요성

Game before post prosessing

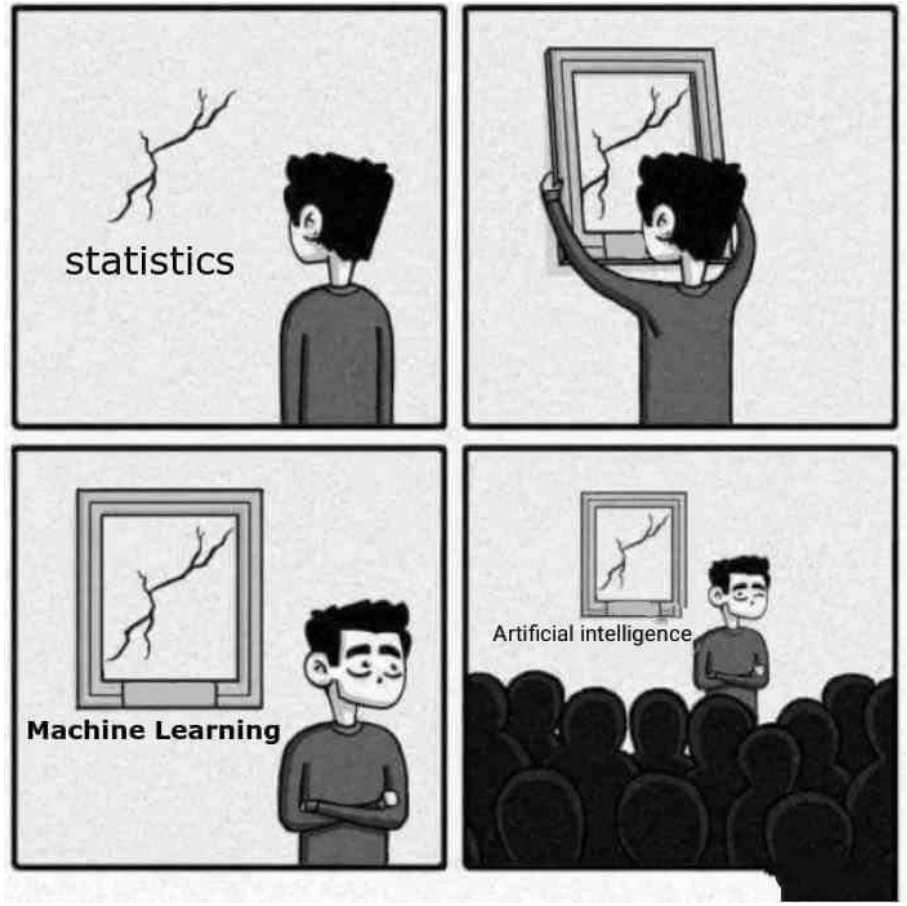
After post prosessing

# 게임 아키텍처 팁..?

- 데이터 지향 설계: 컴포넌트 배치 최적화로 캐시 효율성 20배 향상
- 커스텀 메모리 할당자: 풀링 시스템으로 메모리 단편화 제거
- 고급 상태 관리: 계층적 상태 머신, 행동 트리 활용
- 멀티플레이 장애 복원력: 세션 클러스터링, 상태 지속성, 점진적 기능 저하
- 온라인 게임 표준 아키텍처 구조
- **유저 경험 최우선 원칙**
  - **기술적 완벽함보다 게임플레이 향상에 집중**

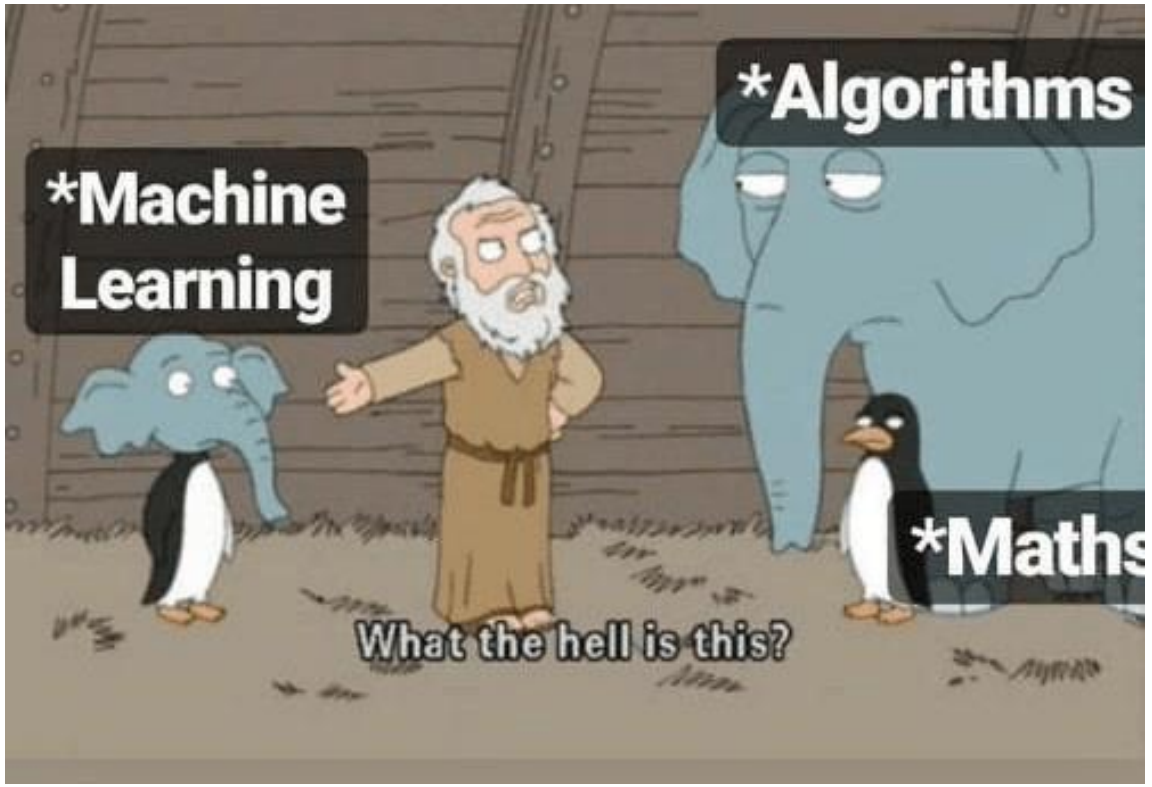
# AI/ML 파이프라인

- 데이터 수집: 초기 로깅 설계에 ML 요구사항 반영 필요
- 실무에서는 데이터의 70-80%가 노이즈, 도메인 지식 중요
- 최신 알고리즘보다 비즈니스 목표 정렬이 중요
- 학문적 정확도보다 비즈니스 가치 측정 필요
- ML 엔지니어와 DevOps 협업 필수
- 모니터링 & 리트레이닝: 데이터 분포 변화에 대응 필요
- GPU/TPU 활용, 분산 학습, 모델 버전 관리의 중요성



# AI/ML 아키텍처 팁..?

- 실패 사례: 수동 학습 및 단일 서버 배포 아키텍처
- 개선된 아키텍처: ETL 파이프라인, Feature Store, 분산 학습, 모델 레지스트리
- 핵심 요소별 실패와 성공 사례
- 구현 시 주의점: 데이터 파이프라인 누수, 리소스 관리, CI/CD 모델 테스트
- 현실적 절충안: 스타트업은 완전한 MLOps보다 관리형 서비스 활용 고려



# 기술 스택 선정 시 고려 사항

- 팀 역량
  - 경험 부족으로 개발 속도 60% 감소 사례
- 프로젝트 요구사항
  - 규모 vs 개발 속도, 성능 요구사항, 실시간성
- 개발/운영 비용
  - 클라우드 vs 온프레미스
  - 오픈소스 vs 상용 솔루션
- 커뮤니티 지원 & 생태계
  - 기술 지속가능성, 인재 풀 고려
- 결정의 가역성: 핵심 기술은 신중하게, 주변 기술은 실험적 접근
- 전략적 의사결정 프레임워크 중요성

# 3주차 과제 (1)

- **기술 스택 및 아키텍처 다이어그램 정리 (03/25 화요일, 23:59 까지) - 노션 페이지 링크 제출**
  - 노션 페이지에 기술 스택 정리해놓을 것.
  - 노션 페이지에 고수준 아키텍처 수준의 다이어그램 원본을 올려놓을 것. (확대해서 볼 수 있게)
    - 웹 프로젝트: 시스템 컨텍스트 다이어그램, 컴포넌트 통신 다이어그램, 배포 개요 다이어그램
    - 게임 프로젝트: 시스템 계층 다이어그램, 핵심 게임 루프 다이어그램, 주요 시스템 간 상호작용
    - 앱 프로젝트: 앱 모듈 구조, 백엔드 통신 흐름, 오프라인/온라인 동작 모델
    - AI/ML 프로젝트: 데이터 흐름 다이어그램, 핵심 파이프라인 개요, 시스템 통합 다이어그램
  - "적응형 아키텍처"의 개념이므로, 설계가 프로젝트 진행에 따라 발전할 수 있어야 됨. 지금은 초기 버전 수준으로 작성.
- **초기 프로젝트 코드 선택한 형상관리 (Git) 툴에 올려 놓을 것. (03/25 화요일, 23:59 까지) - 링크 제출**
  - 기본 구조, “Hello World” 수준이면 됨.

# 3주차 과제 (2)

- **웹 프로젝트**
  - **시스템 컨텍스트 다이어그램**: 사용자, 외부 시스템, 주요 인터페이스를 포함한 전체 시스템 조망
  - **컴포넌트 통신 다이어그램**: 프론트엔드, 백엔드, 데이터베이스, 외부 API 간의 주요 통신 흐름
  - **배포 개요 다이어그램**: 서버/클라이언트 구조와 주요 인프라 요소
- **게임 프로젝트**
  - **시스템 계층 다이어그램**: 엔진, 게임 로직, UI, 입출력 시스템 등의 주요 계층
  - **핵심 게임 루프 다이어그램**: 주요 게임 상태와 상태 전이를 보여주는 흐름도
  - **주요 시스템 간 상호작용**: 물리, AI, 렌더링, 사운드 등 주요 시스템의 연결

# 3주차 과제 (3)

- **AI/ML 프로젝트**
  - **데이터 흐름 다이어그램**: 데이터 소스에서 최종 출력까지의 고수준 흐름
  - **핵심 파이프라인 개요**: 데이터 수집, 전처리, 모델링, 평가, 배포의 주요 단계
  - **시스템 통합 다이어그램**: AI/ML 모델이 다른 시스템과 어떻게 통합되는지 표현
- **모바일 앱 프로젝트**
  - **앱 모듈 구조**: 주요 기능 모듈과 그 관계
  - **백엔드 통신 흐름**: 앱-서버 간 API 통신과 데이터 흐름
  - **오프라인/온라인 동작 모델**: 네트워크 상태에 따른 앱 동작 방식

# Next Topic

## 4주차. 스프린트 1 - 핵심 기능 개발 I