

TAL Sprint1 Progress

Sprint1 목표

- 1st 멘토링 진행
- 기획 방향성 확정 및 수정
- SDK 개발 및 비교
- 디자인 및 컨셉 확정
- 제스처 측면

작성 문서 목록

해당 문서들은 [takealook-docs](#)의 sprint1 폴더 안에 작성

1. team02-Proposal-Report-Improve : proposal report 방향성을 고려하여 수정
2. TAL 멘토링 보고서 : 3/30 진행한 멘토링 보고서 (해당 일자의 진행상황 및 멘토링 피드백 내용 포함)

피드백 반영

- TAL의 주기능을 자세 교정으로 확정

팀원과의 의견 차이가 있었고 많은 회의를 진행하여 4/7에 진행한 회의로 확정

제스처를 최소화하는 이유

1. 제스처를 위하여 목을 신경쓰고 고정하고 움직이는 것이 불편
2. 제스처의 다양성을 확보 불가 (8방향 이외 제스처의 부재)

- 구현 가능성을 고려하지 않은 브레인 스토밍 (관련 자료는 [TAL 회의 자료](#) 참조)

자세 교정 관련 피드백을 제시할 요소 확정

1. 거북목
2. 고개 기울어짐, 턱 고기, 다리 고기 → 어깨 불균형
3. 장시간 착석
4. 눕듯이 앉기

- 강제성에 대한 우려

강제성을 지닌 피드백은 생산성을 낮춤과 동시에 사용자에게 불쾌함을 제공

→ 강제성을 지니지만 불쾌하지 않은 키치하고 유쾌한 방식으로 피드백 기획

기획 (UX 및 문서화)

건강관리 vs 생산성

처음 기획 방향은 제스처였지만 건강관리 측면에서 접근으로 기획을 수정
따라서 제스처를 최소화한 자세 인식 및 피드백에 집중하여 목표 변경
해당 내용은 [Improve Proposal Report](#)에 제공

변경 사항은 파란색, 제외 사항은 ~~취소선~~으로 구분

자세 교정 피드백

이전 확정된 자세 교정 요소에 대한 피드백 구체화
모든 자세 교정 피드백은 측정한 특정 수치가 넘어가면 실행되도록 제작
이 수치는 사용자가 조절하여 피드백의 민감도를 설정 가능
또한 각 항목들을 켜거나 끌 수 있음.

거북목

고개가 앞으로 쏠리는 정도로 구분
고양이 모션 (고양이의 행동을 기반으로 한 피드백 제공)
ex. 꼬리로 모니터 일부 가리기, 스크래치 남기기, 훑기 등

다른 후보

- 글자 확대
- 모니터에 다른 화면 띄우기

어깨 불균형

턱 괴기 혹은 다리 꼬기와 같은 요소로 인하여 어깨가 불균형 상태일 때 피드백
기울어진 구석에 고양이 낙하 및 방치
고양이가 기울어진 쪽에 고양이 모델이 떨어지며 올바른 자세로 돌아가기 전까지 사라지지 않음.

다른 후보

- 화면 자체를 기울어지게 출력
- 기울어진 쪽 화면 가리기

장시간 착석

장시간 앉아 있는 것은 부정적인 역할을 줄 수 있기 때문에 1시간마다 30초~1분 정도의 스트래칭을 유도
화면을 블러 처리하고 30초 동안 스트래칭 권장

잠시 일어나서 허리를 펴주세요

위와 같은 문장 출력

눕듯이 앓기

가슴, 목, 얼굴의 가상의 선을 분석하여 평소보다 허리를 앞으로 뺀 눕는 자세를 인식

현재 피드백을 기획 X

SDK

swift 내장 라이브러리, openCV로 개발 중

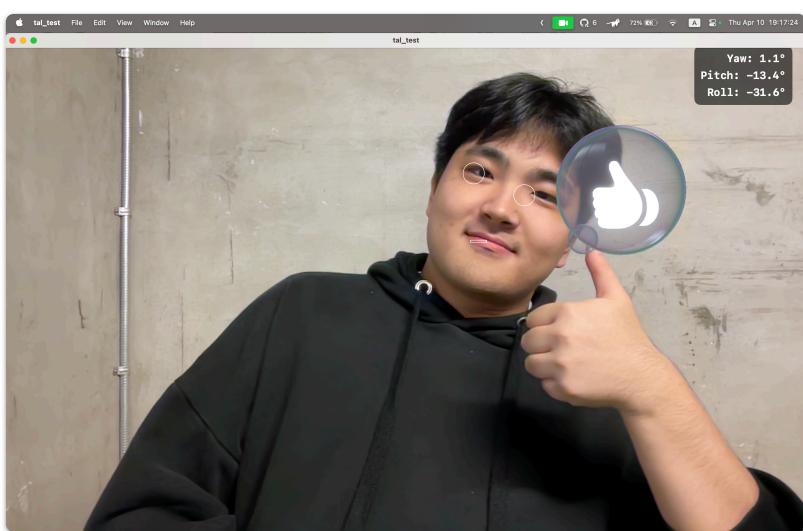
언어	속도	지원 기능	SDK 연동	비고
Swift	빠름	고개 각도만 측정 가능	매우 편리 (iOS 환경 최적화)	Swift 내장
Python	느림	고개 각도 + 어깨 각도	Swift SDK 연동 불편	OpenCV 기반
C++	빠름	고개 각도 + 어깨 각도	연동 가능	OpenCV 기반

현재 Swift와 Python으로 구현

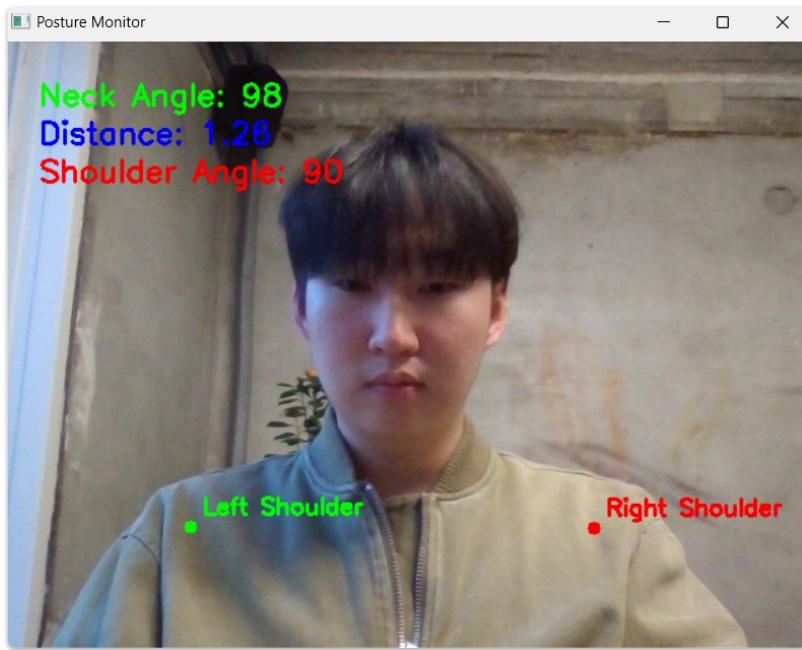
- swift로 개발하였지만 어깨 각도 측정 불가
- python은 sdk로 사용하기 위해선 coreML과 같은 sdk 형식을 다르게 변환하거나 서버를 활용
→ 서버는 로깅 및 확장성 면에서 좋지만 네트워크 연결 및 서버의 리소스를 초과할 가능성 존재

작은 리소스를 목적으로 하기 때문에 딥러닝 모델을 배제하여 순수 openCV 모델로만 제작 예정

python으로 우선 제작하여 sdk 연결에 실패하면 c++로 리라이팅하여 제작



위 사진은 swift로 구현한 웹캠 얼굴 각도 측정 (Yaw, Pitch, Roll)



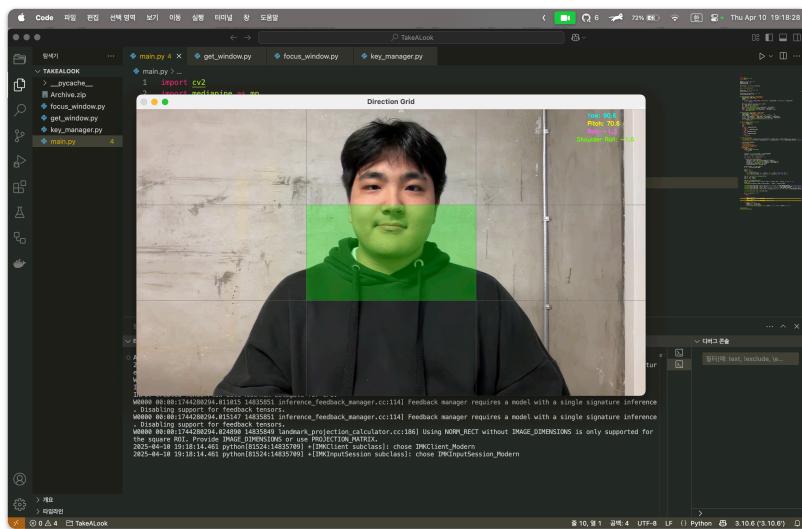
python을 사용하여 구현한 웹캠을 실시간으로 분석한 사진

제스처 측면

python으로 sdk를 제작

얼굴 방향을 9분할하여 인식

기획 수정으로 더 디벨롭 x



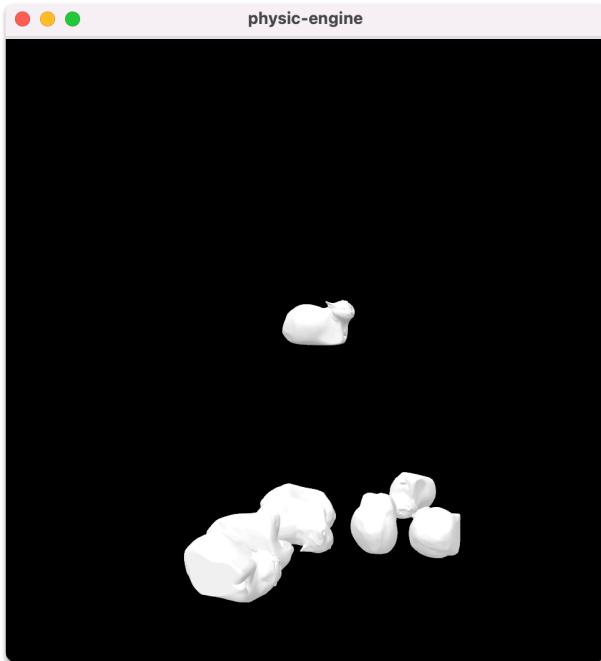
위 사진은 python opencv로 구현한 제스처 인식

Frontend

2D vs 3D

3D

고양이가 떨어지고 쌓이는 파트에서 3D 모델링된 고양이를 사용



swift로 물리엔진을 구현하여 떨어지고 충돌하는 3D 뷰 구현

1. UI와 조화가 좋지 못함 → 다른 요소에서 3D를 활용 x
2. 리소스 과소비 (m2 노트북 기준 20~30%)

2D

픽셀을 고려했었지만 너무 아케이드적 요소가 결합될 여지가 존재

→ 모던한 고양이 일러스트를 기반으로 제작



위 캐릭터를 활용하여 아이콘 및 피드백 기능 제공

menu bar

mac os의 menu bar로 간단하게 설정 및 통계를 확인 가능



위 사진은 현재 구현한 swift 기반 menu bar application