

WebXR API를 이용한 Web AR 개발, 그 한계와 대안

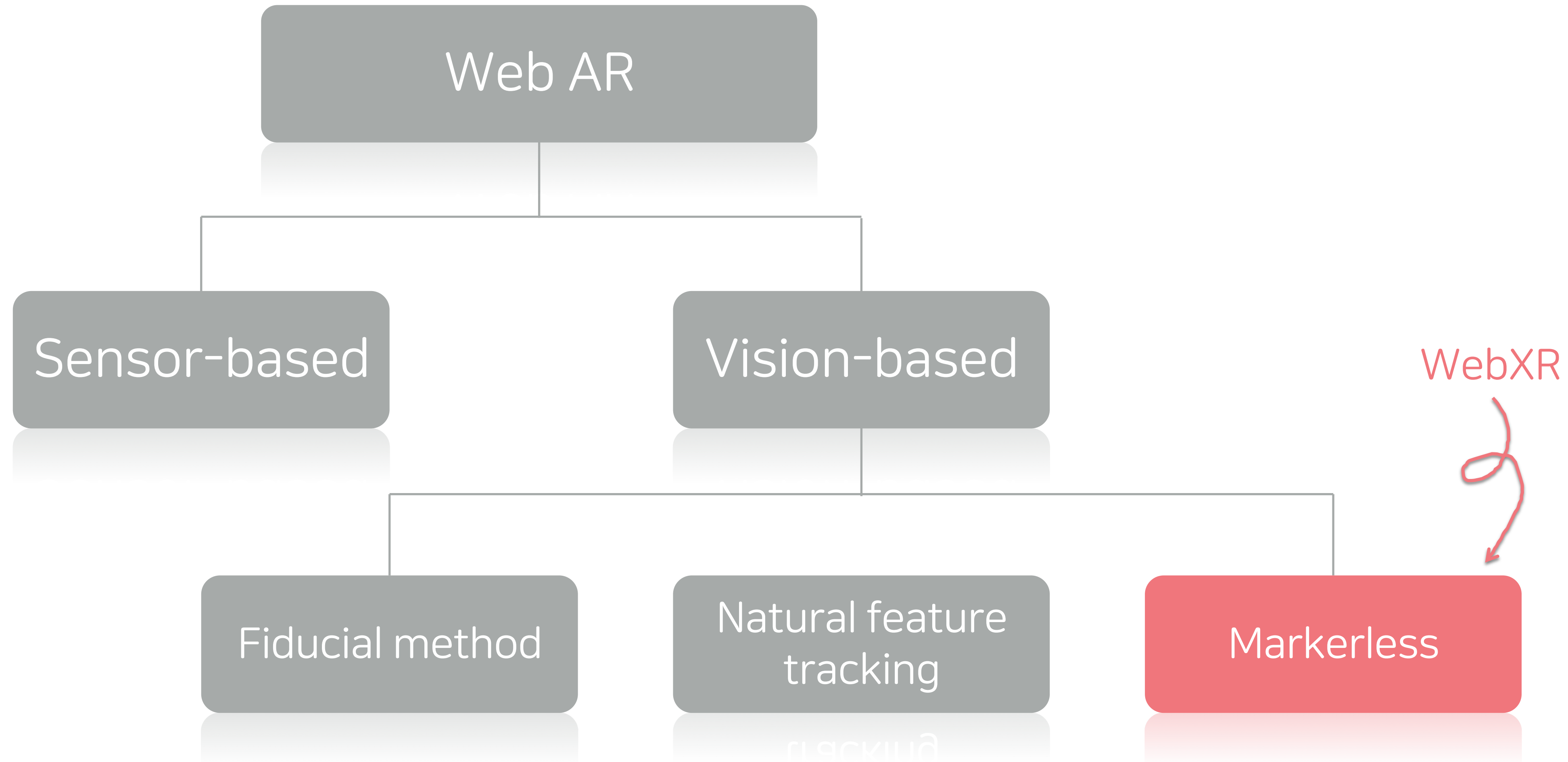
김민규 NAVER FE Platform

CONTENTS

1. Web AR Approaches
2. Web AR with WebXR API
3. WebXR의 한계와 대응
4. 적용시 고려사항

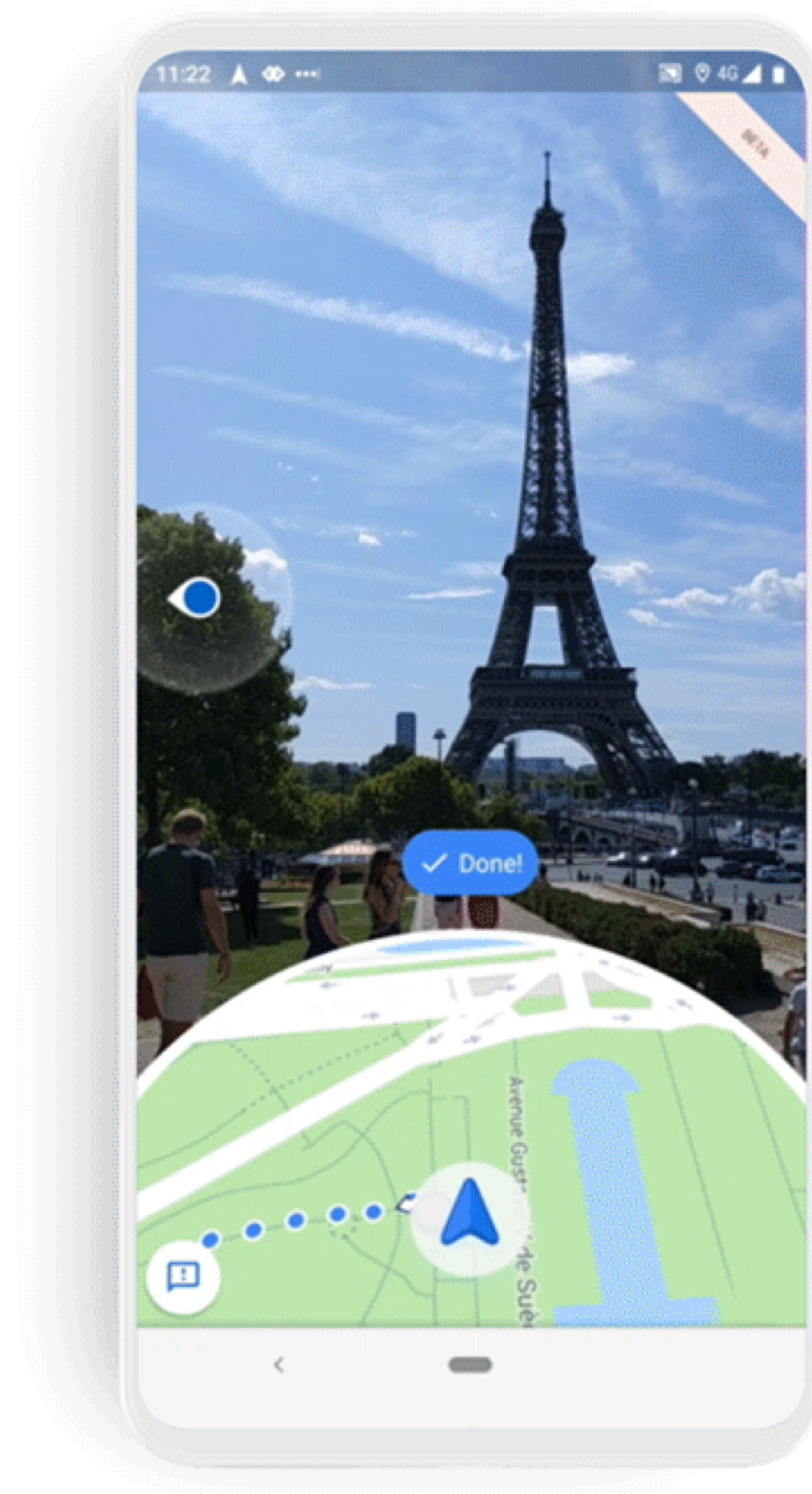
1. Web AR Approaches

1. WebAR Approaches



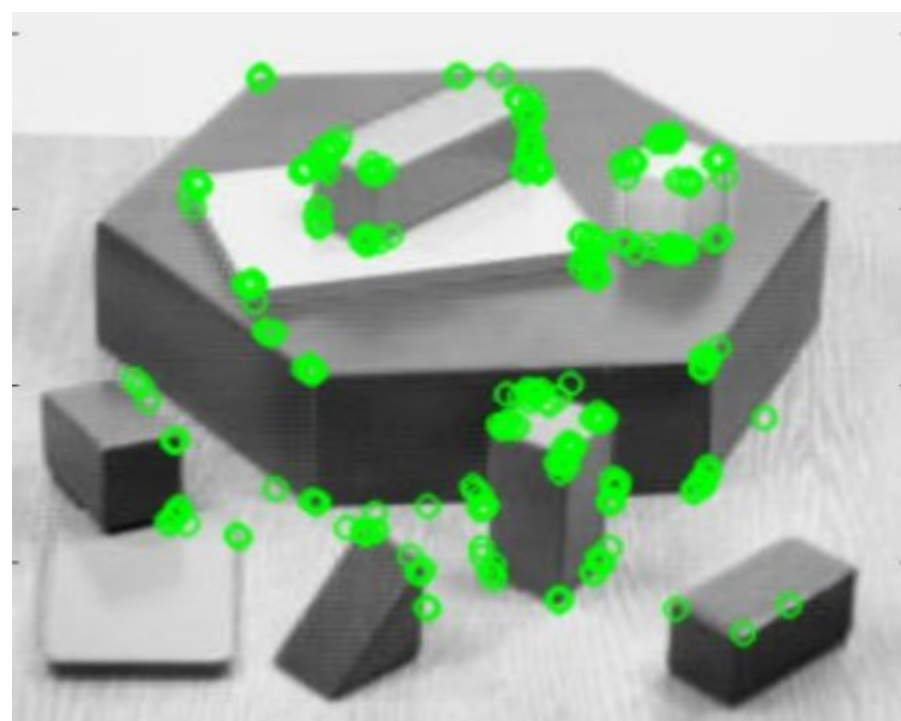
웹 AR 구현 방식들

1.1 Sensor-based AR

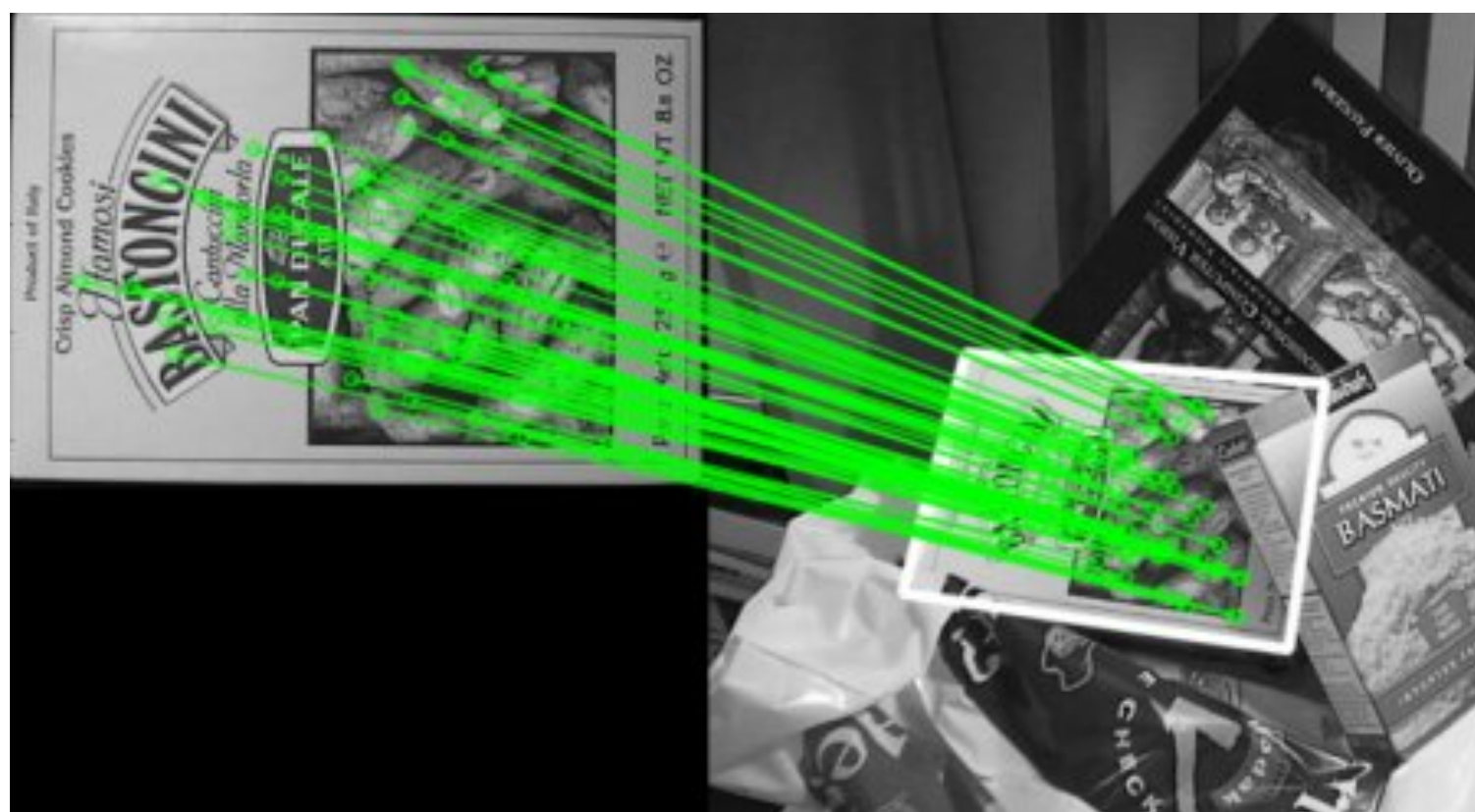


- 가속도계, 자기장계, GPS 등의 센서를 이용한 AR
- 비전 기반 방식에 비해 상대적으로 적은 연산량
- 센서의 성능에 크게 영향받고, 오차가 누적됨(open-loop)
- 비전 방식과 결합한 하이브리드 방식으로 사용되기도 함

1.2 Vision-based AR



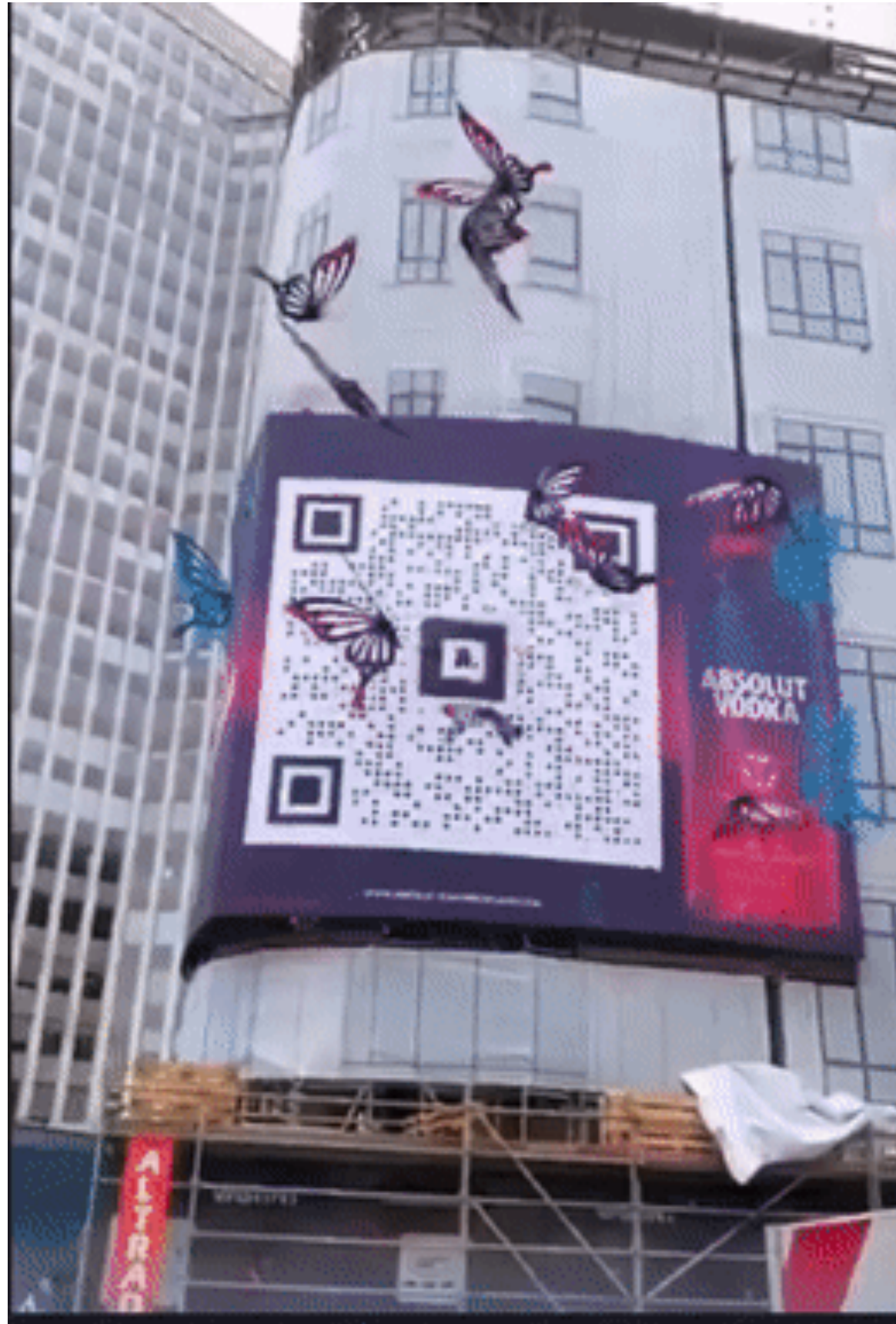
- Feature Detection(특징점 추출) 및 Matching에 기반
- 모바일 디바이스 - 카메라 피드를 이용
- Frame by Frame(closed-loop)
- 오차 누적 없음
- 높은 연산 능력 필요



Images from OpenCV,

https://docs.opencv.org/3.4/db/d27/tutorial_py_table_of_contents_feature2d.html

1.2 Vision-based AR



Demo image from ar.js, <https://ar-js-org.github.io/AR.js-Docs/>

Fiducial Method

- 일정한 패턴을 가진 이미지(마커)를 Tracking
- 정확도가 높고, 환경 변화에 크게 영향받지 않음
- 마커가 필요하기 때문에 제한적으로 적용할수밖에 없음

1.2 Vision-based AR

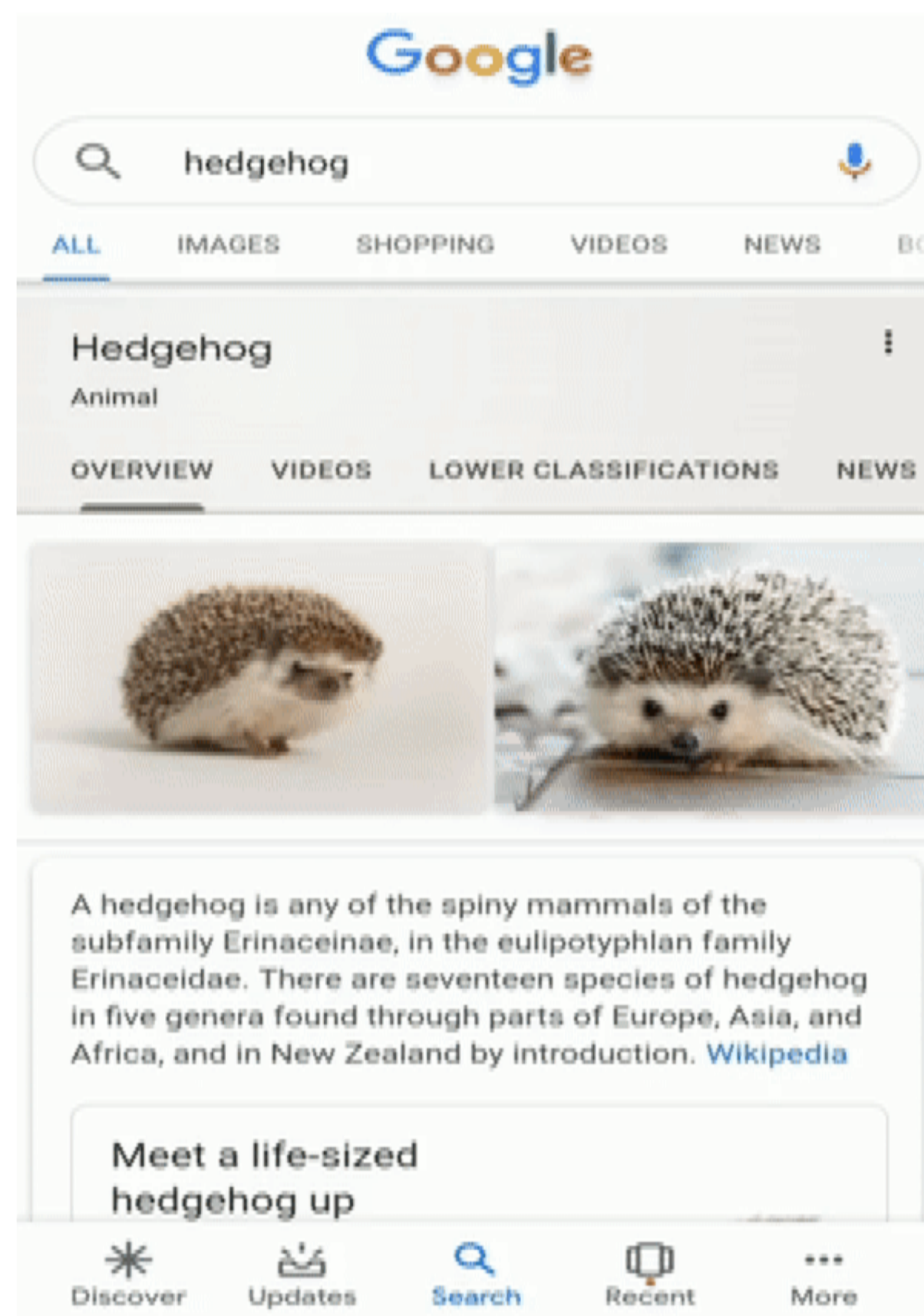


Demo image from ar.js, <https://ar-js-org.github.io/AR.js-Docs/>

Natural Feature Detection

- 이미지를 인식하여 Tracking
- Feature Detection이 가능한 이미지만 인식
- 얼굴인식 등 3D 오브젝트 인식도 포함
- 마커 이용 방식에 비해 더 높은 인식 성능을 필요로 함

1.2 Vision-based AR

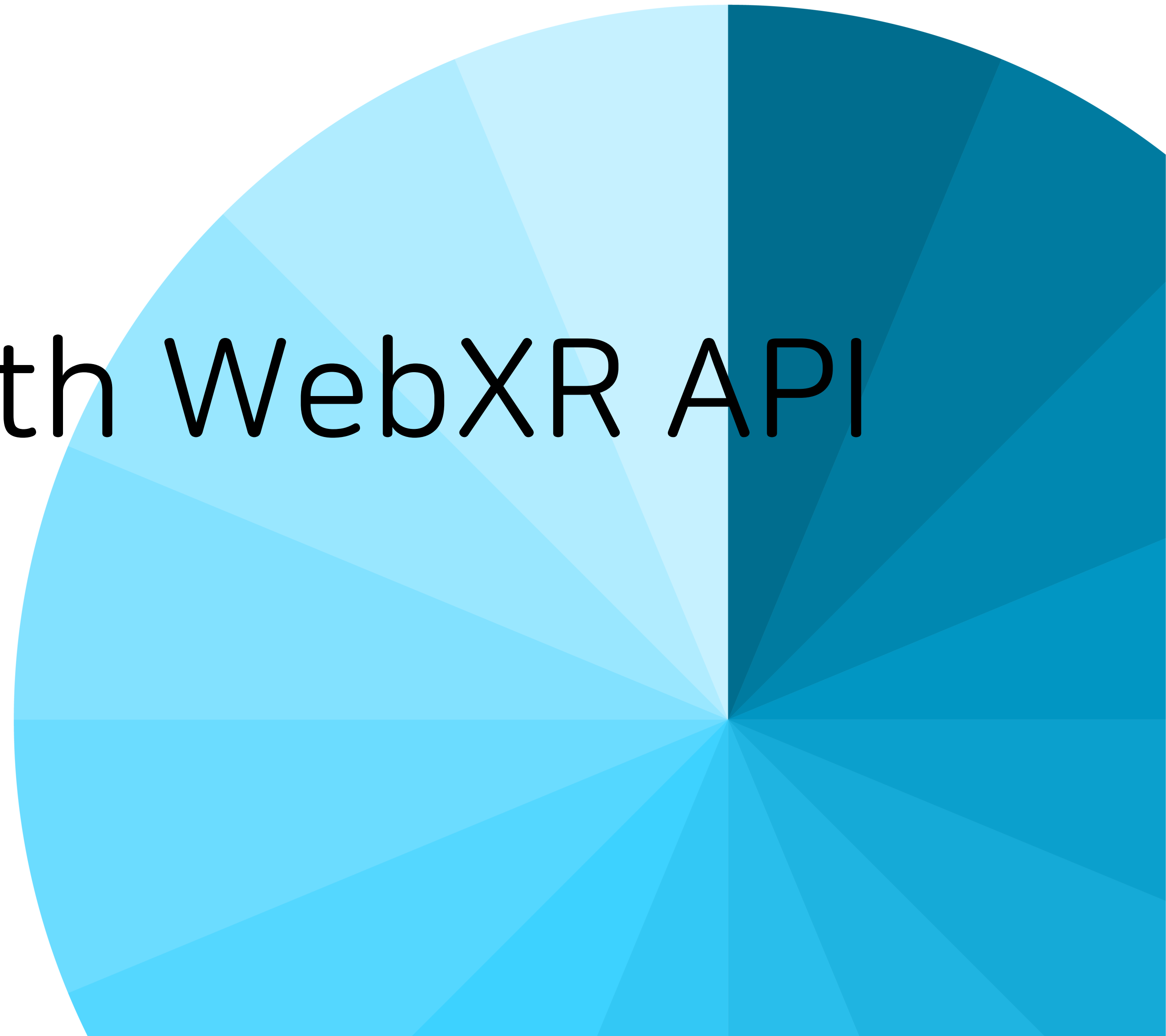


Demo image from Google AR, <https://arvr.google.com/ar/>

Markerless

- 사전 정보 없이 Motion Tracking / 주변 환경을 인식
- SLAM(Simultaneous Localization and Mapping)
- 높은 연산량
 - Sensor-based 방식과 결합, Hybrid방식으로 사용
- 자체 SLAM 엔진을 갖출 경우 최신 API인 WebXR에 비해 더 높은 브라우저 커버리지
 - WebRTC, WebAssembly, WebGL, Web Worker, ...

2. Web AR with WebXR API



2.1 Before WebXR



WebXR API 발전사

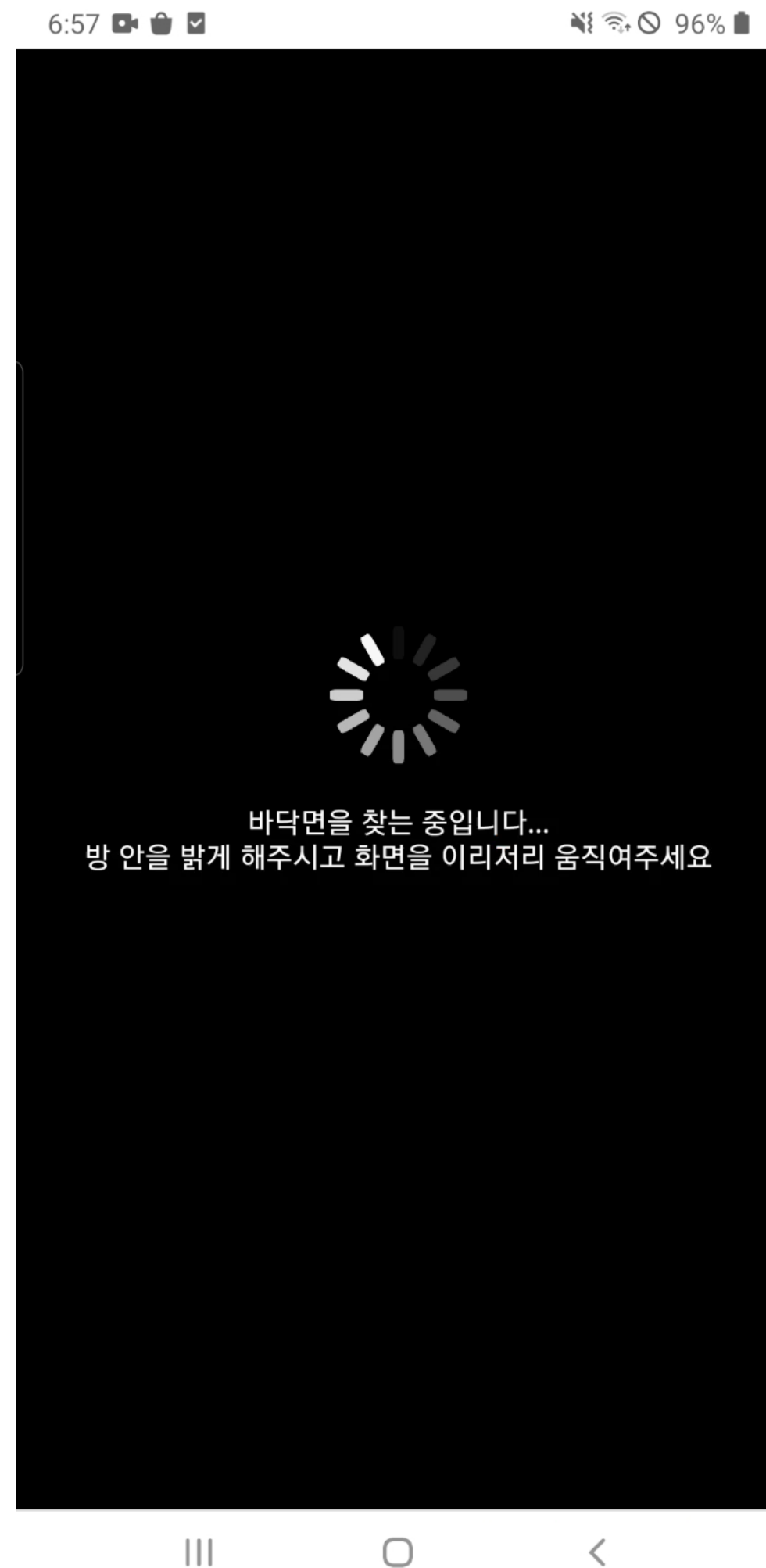
2.1 Before WebXR



Demo image from WebARonARCore, <https://github.com/google-ar/WebARonARCore>

- Google의 WebARonARCore / WebARonARKit
- ARCore(Android) / ARKit(iOS) 기반
- Chromium / WKWebView 확장
- WebVR API 1.1을 확장하여 AR기능을 추가

2.2 WebXR API



- 기존 WebVR API를 deprecated시키고 계승
- 현재 Chromium-based 브라우저에만 적용
 - ARCore(Android)를 기반
- Markerless
 - SLAM(Simultaneous Localization and Mapping)
- Experimental - API 변경 가능성 존재

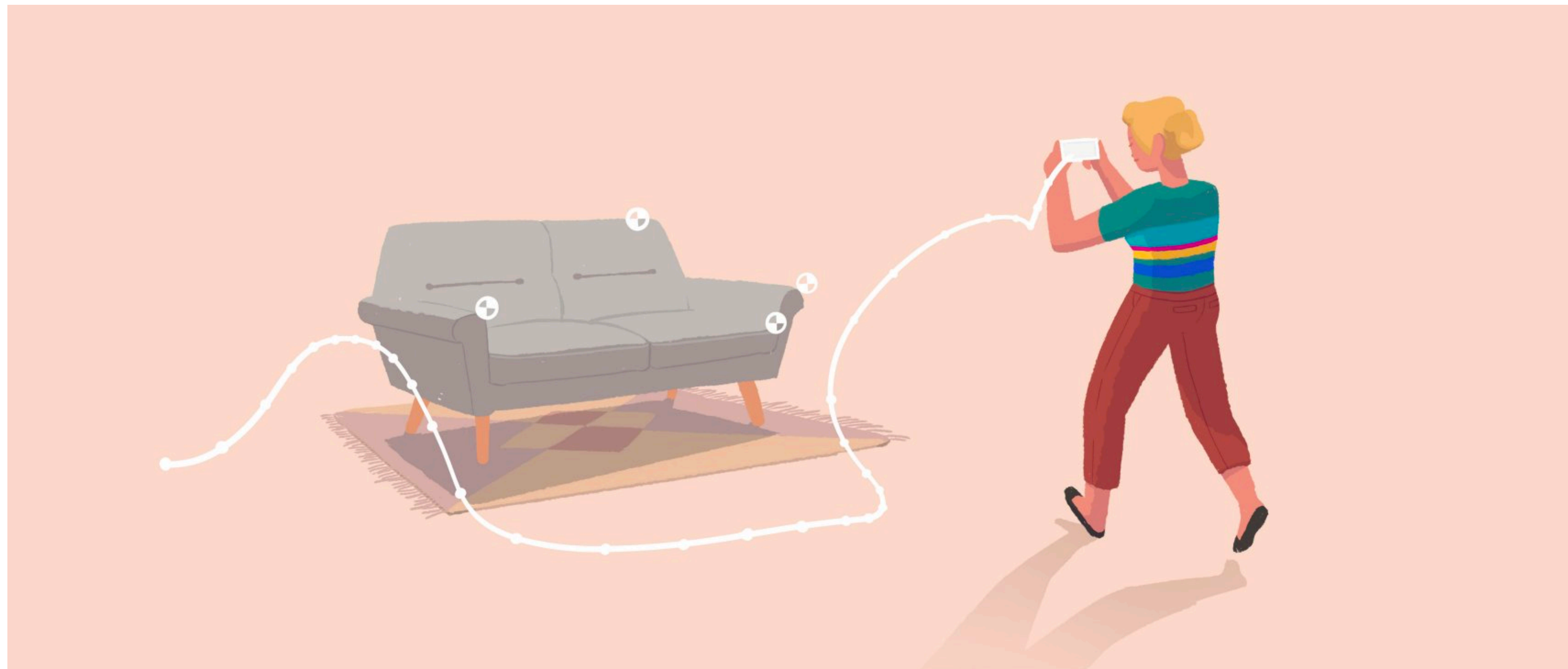
2.3 WebXR API – Features

```
navigator.xr.requestSession("immersive-ar", {  
  requiredFeatures: ["hit-test"],  
  optionalFeatures: ["dom-overlay"],  
});
```

- WebXR 세션에서 사용할 기능을 지정
- immersive-web Github에서 논의중인 feature들 확인 가능
 - Lighting-estimation, depth-sensing, environment-blending ...
- <https://github.com/immersive-web/>

2.3 WebXR API – Features

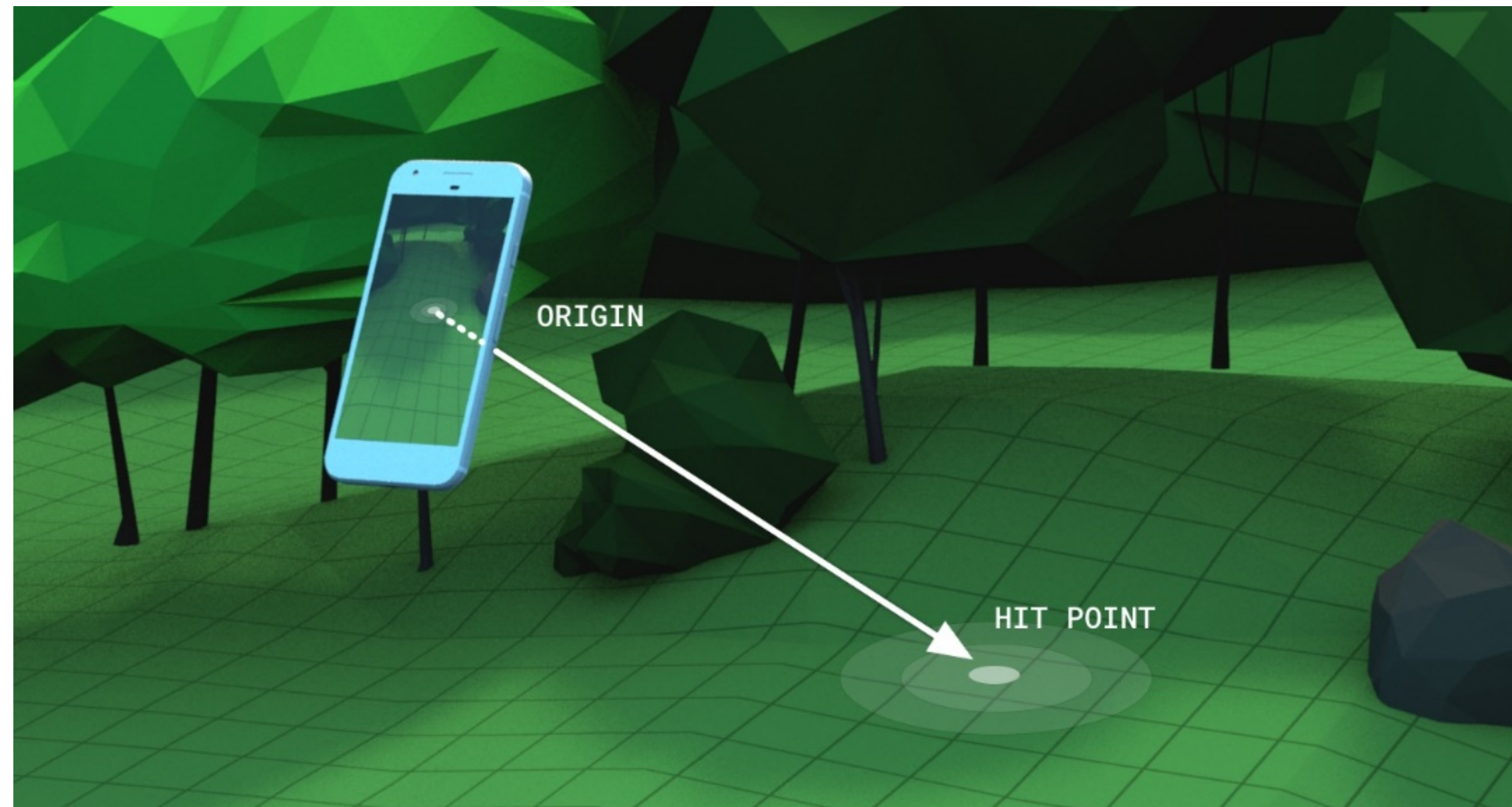
Motion tracking



<https://developers.google.com/ar/discover/concepts>

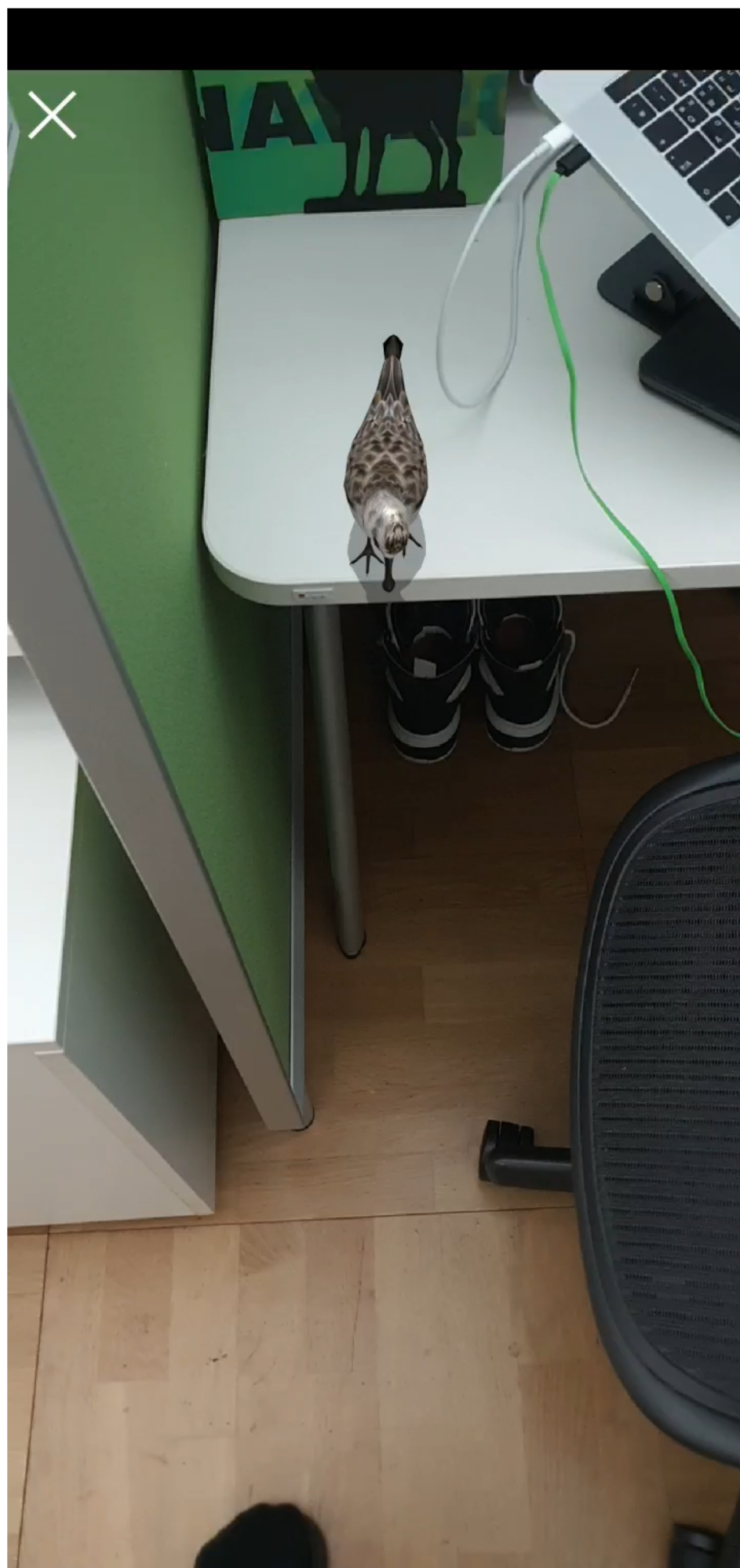
2.3 WebXR API – Features

hit-test



<https://codelabs.developers.google.com/codelabs/ar-with-webxr/#4>

2.3 WebXR API – Features



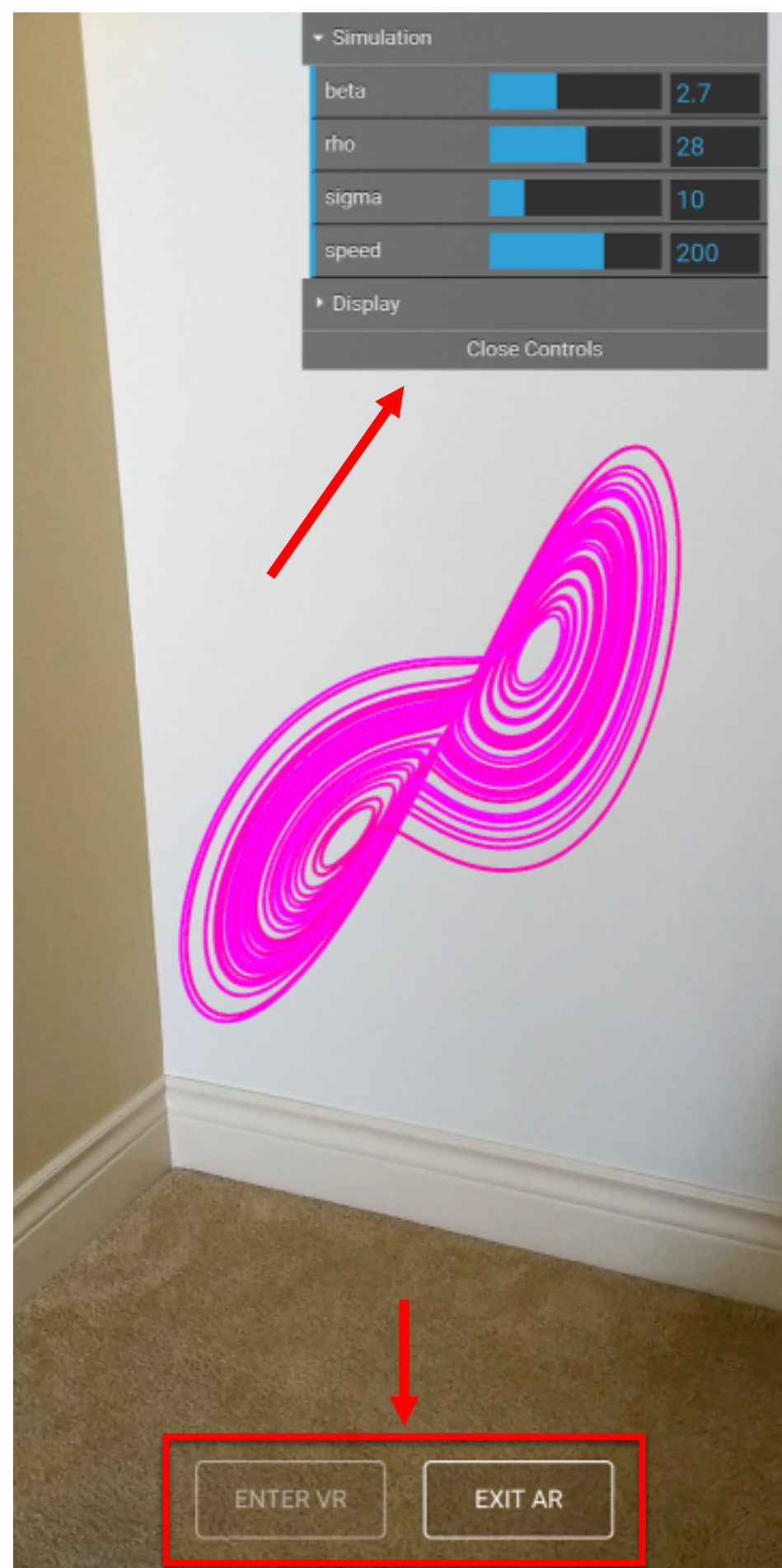
```
// Retrieve transient hit-test source
session.requestHitTestSourceForTransientInput({ profile: "generic-touchscreen" })
  .then((transientHitTestSource: any) => {
    this._hitTestSource = transientHitTestSource;
  });

// ...

// Get hit-test results from frame using previous hit-test source
const hitResults = frame.getHitTestResultsForTransientInput(this._hitTestSource);
```

- 사용자의 입력을 기준으로도 hit-test 수행가능
- TransientInput

2.3 WebXR API – Features

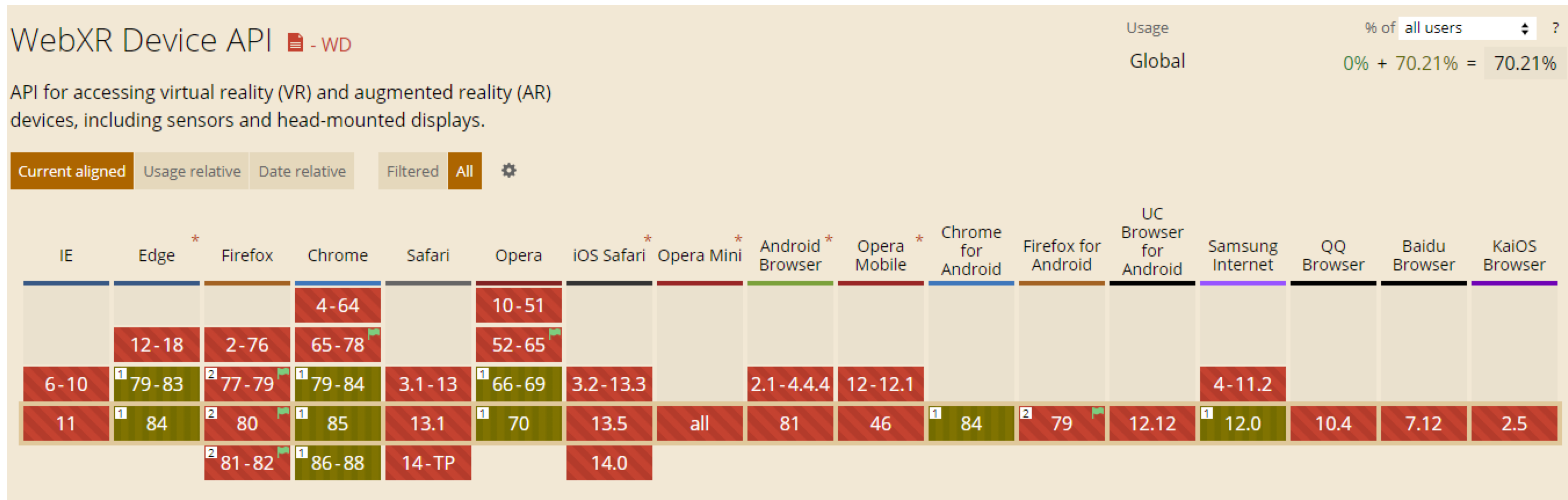


dom-overlay

- XR 세션위에 DOM 엘리먼트 표시
- 웹의 다른 콘텐츠들을 오버레이 형태로 표시 가능
- UI 제작 / CSS3 3D transform 이용하여 웹 콘텐츠를 실제 공간에 표시

3. WebXR의 한계와 대응

3.1 WebXR – Browser Coverage



<https://caniuse.com/webxr>

굉장히 낮은 브라우저 커버리지(Chrome 81+)

3.1 WebXR – Device Coverage

ARCore supported devices

[Send](#)

What does it mean for a device to be supported? In short, it means it has passed our certification process.

- “Quality of the camera, motion sensors, and the design architecture to ensure it performs as expected
- “The device needs to have a powerful enough CPU that integrates with the hardware design to ensure good performance and effective real-time calculations.

3.2 WebXR – Polyfill

WebXR Polyfill

build failing npm v2.0.3

A JavaScript implementation of the [WebXR Device API](#), as well as the [WebXR Gamepad Module](#). This polyfill allows developers to write against the latest specification, providing support when run on browsers that implement the [WebVR 1.1 spec](#), or on mobile devices with no WebVR/WebXR support at all.

The polyfill reflects the stable version of the API which has shipped in multiple browsers.

- immersive-web/webxr-polyfill
- 기존 webvr-polyfill을 WebXR API 스펙에 맞춘 형태
- VR 세션("immersive-vr")만을 지원

3.3 WebXR – Fallback(Android)



<https://arvr.google.com/ar/>

- WebXR API가 존재하지 않는 브라우저를 커버
- Google의 SceneViewer 사용
 - com.google.android.googlequicksearchbox
 - com.google.ar.core
- glTF 포맷 사용(Draco 지원)
- 3d / AR 모드가 따로 존재
- Intent기반 앱 호출

3.3 WebXR – Fallback(iOS)



<https://www.apple.com/kr/newsroom/2018/06/apple-unveils-arkit-2/>

- Apple의 AR QuickLook
- iOS11+ 디바이스를 지원
- usdz/reality 포맷 사용
- <a>태그 기반

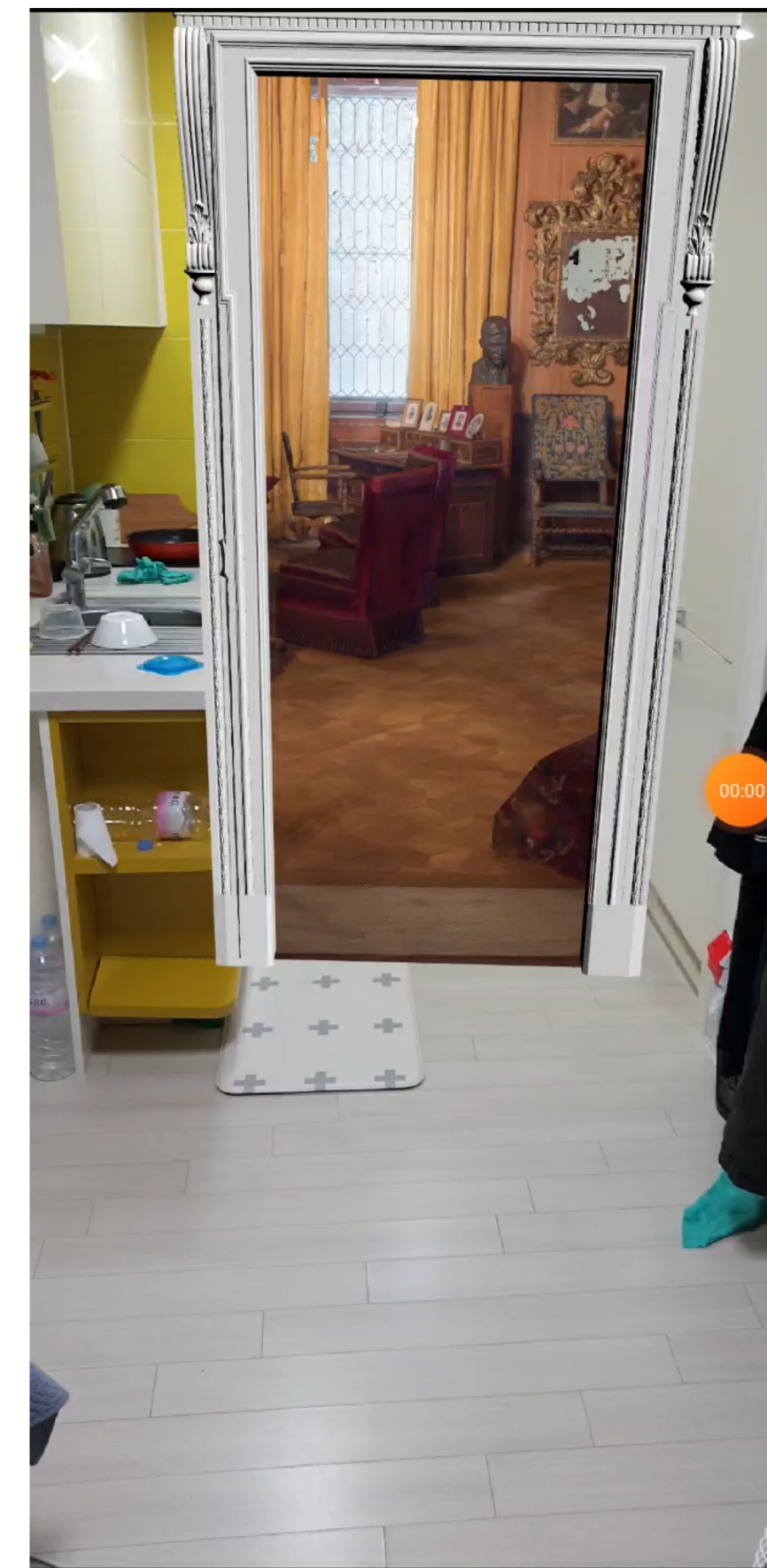
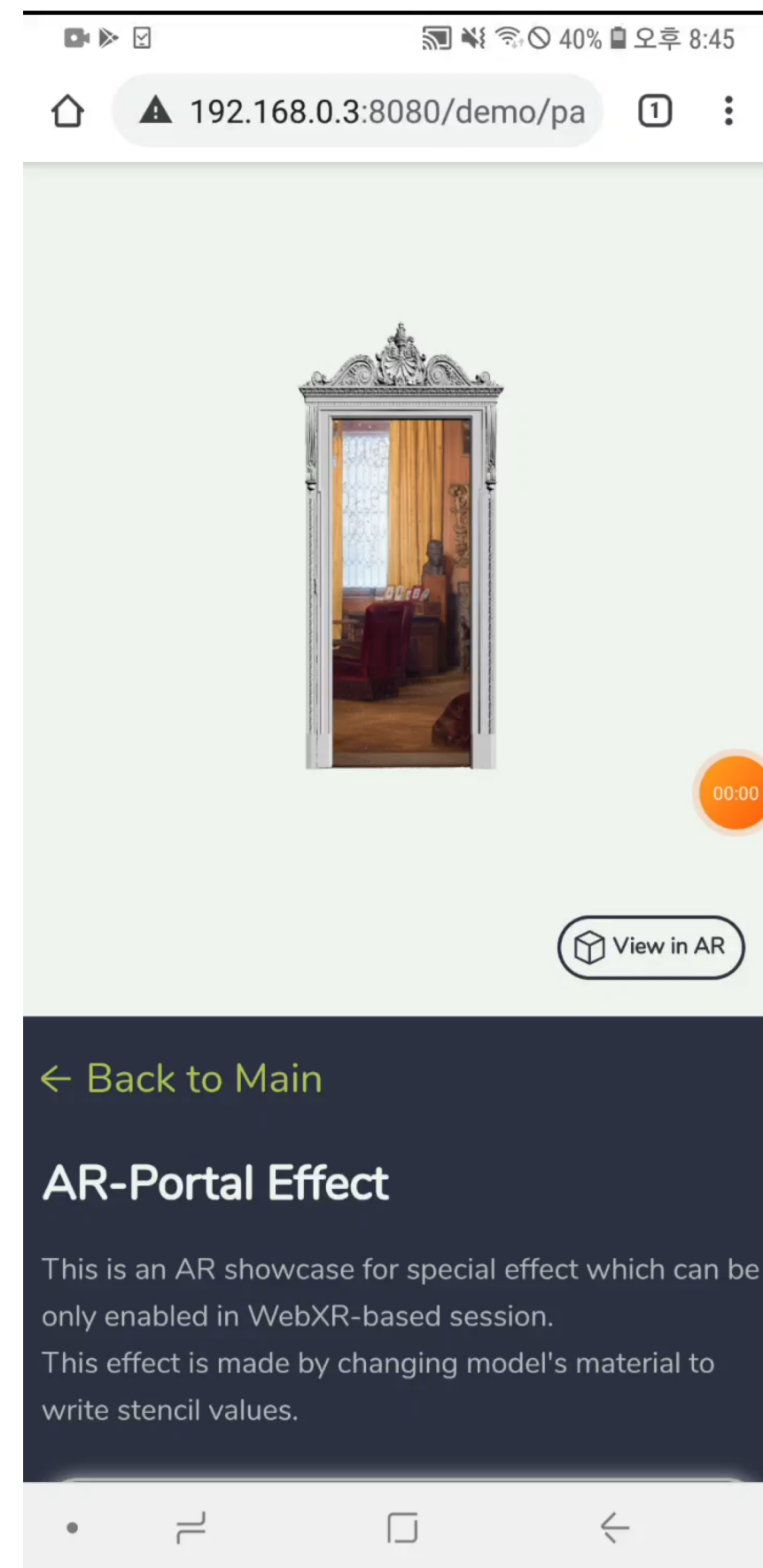
```
<a rel="ar" href="/assets/models/my-model.usdz">  
    
</a>
```


3.3 WebXR – Fallback에선 불가능한 것들



쥬니어네이버 AR – 앱 이외 인터랙션 불가능

3.3 WebXR – Fallback에선 불가능한 것들



3D 모델로는 저장 불가능한 Special Effect들 활용 불가

3.4 WebXR - 기능적 한계



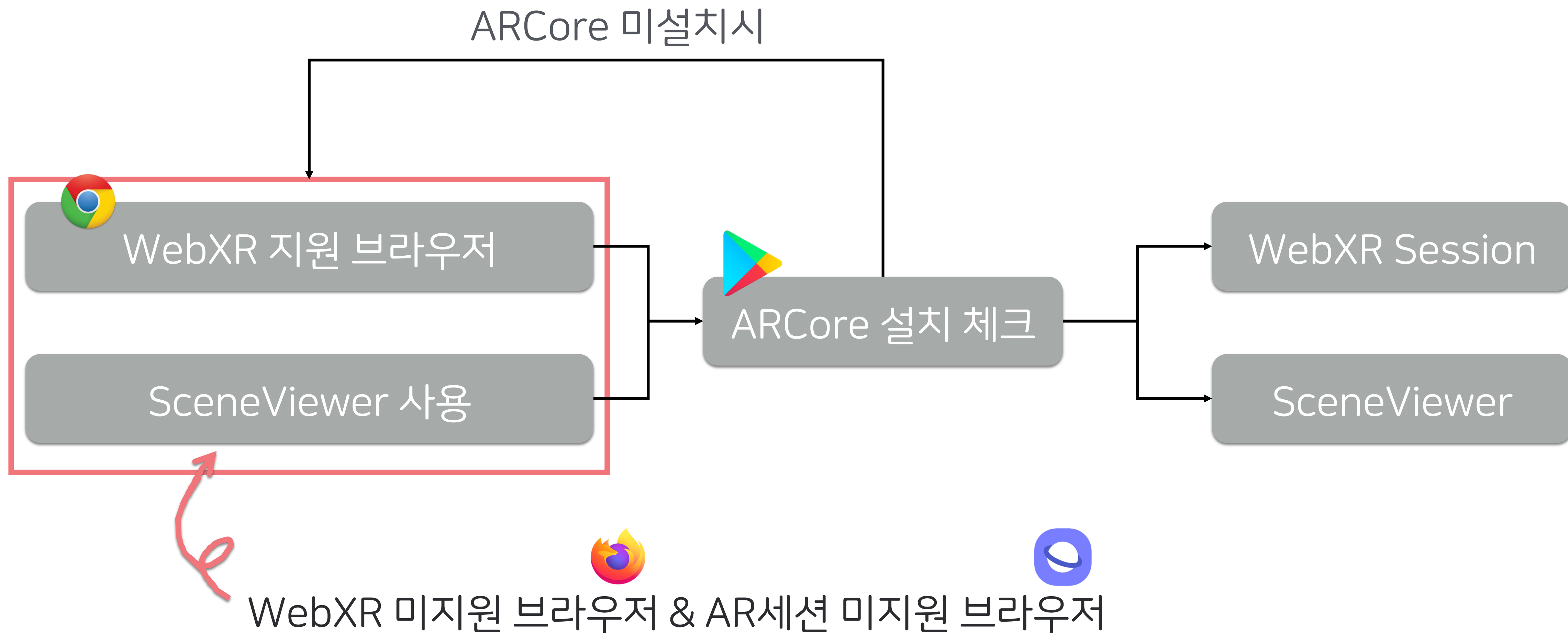
Environment blending

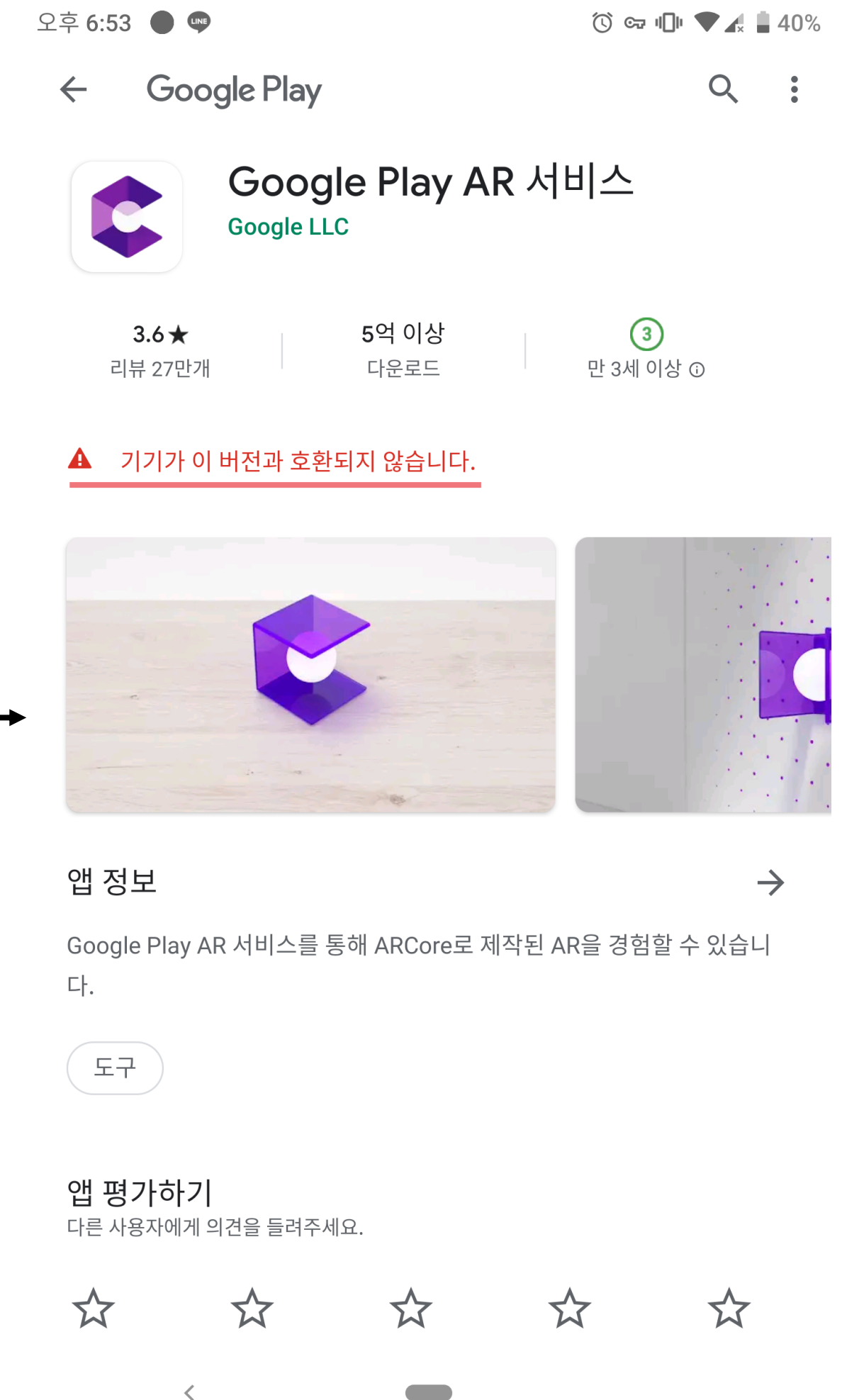


Surface detection

현재로서는 ARCore의 기능들을 전부 지원하고 있진 않음
실제로 어떤 기능을 사용해야 하는지 파악하는 것이 필요

3.5 Flow – Android (ARCore 지원)



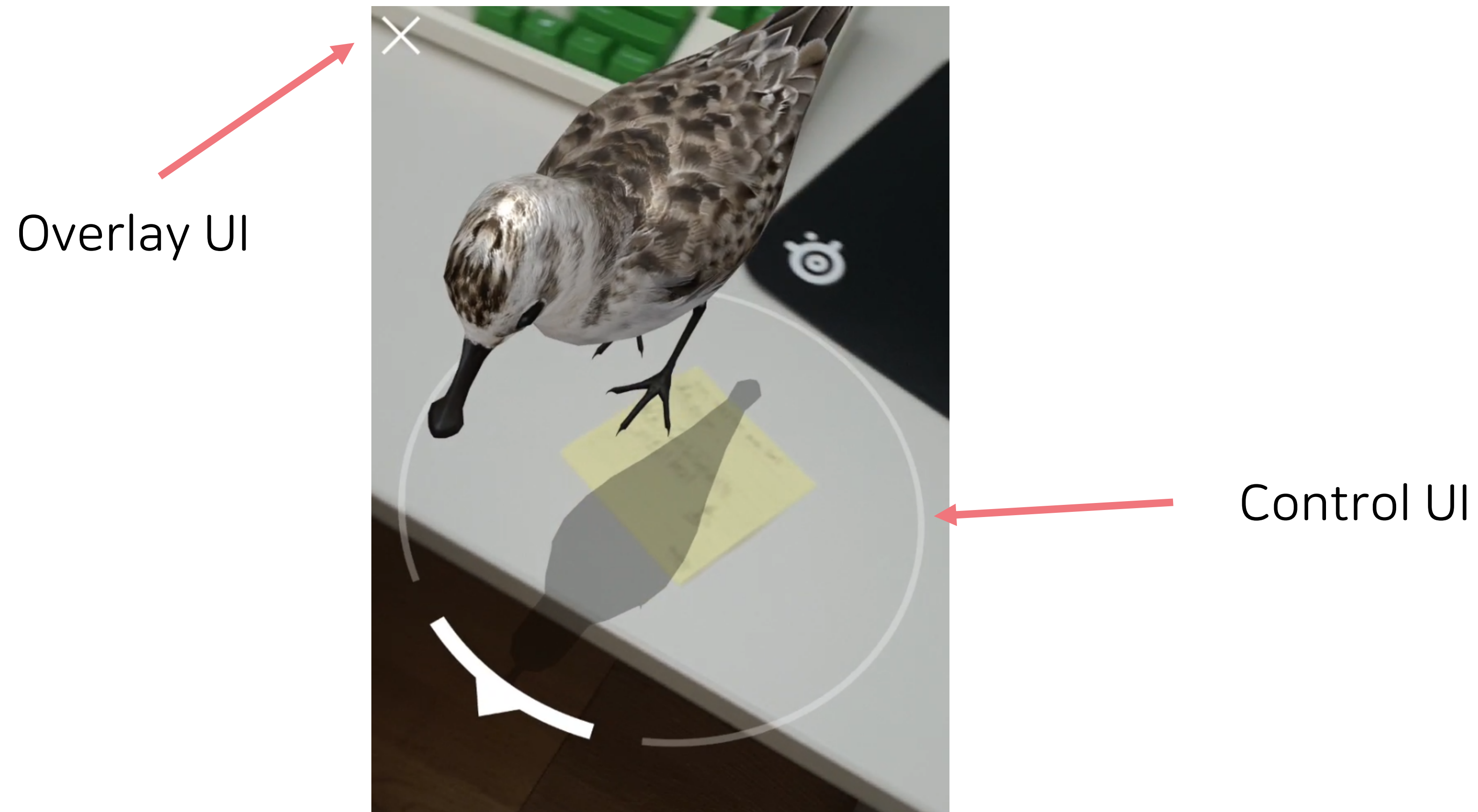


3.5 Flow – iOS11+



4. 적용시 고려사항

4.1 구현해야 하는 것들

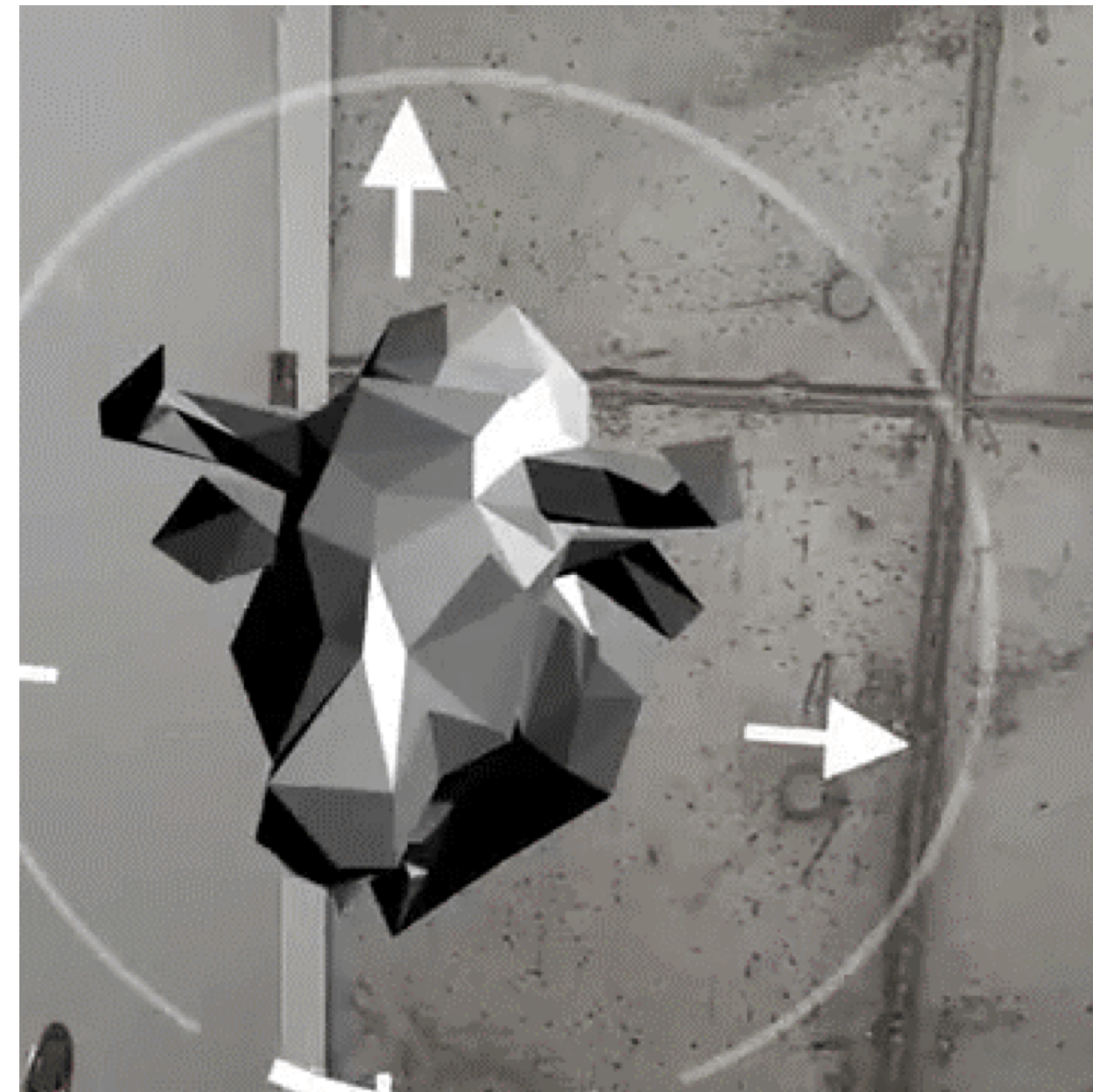


이외에 컨트롤, 모델 표시 등...

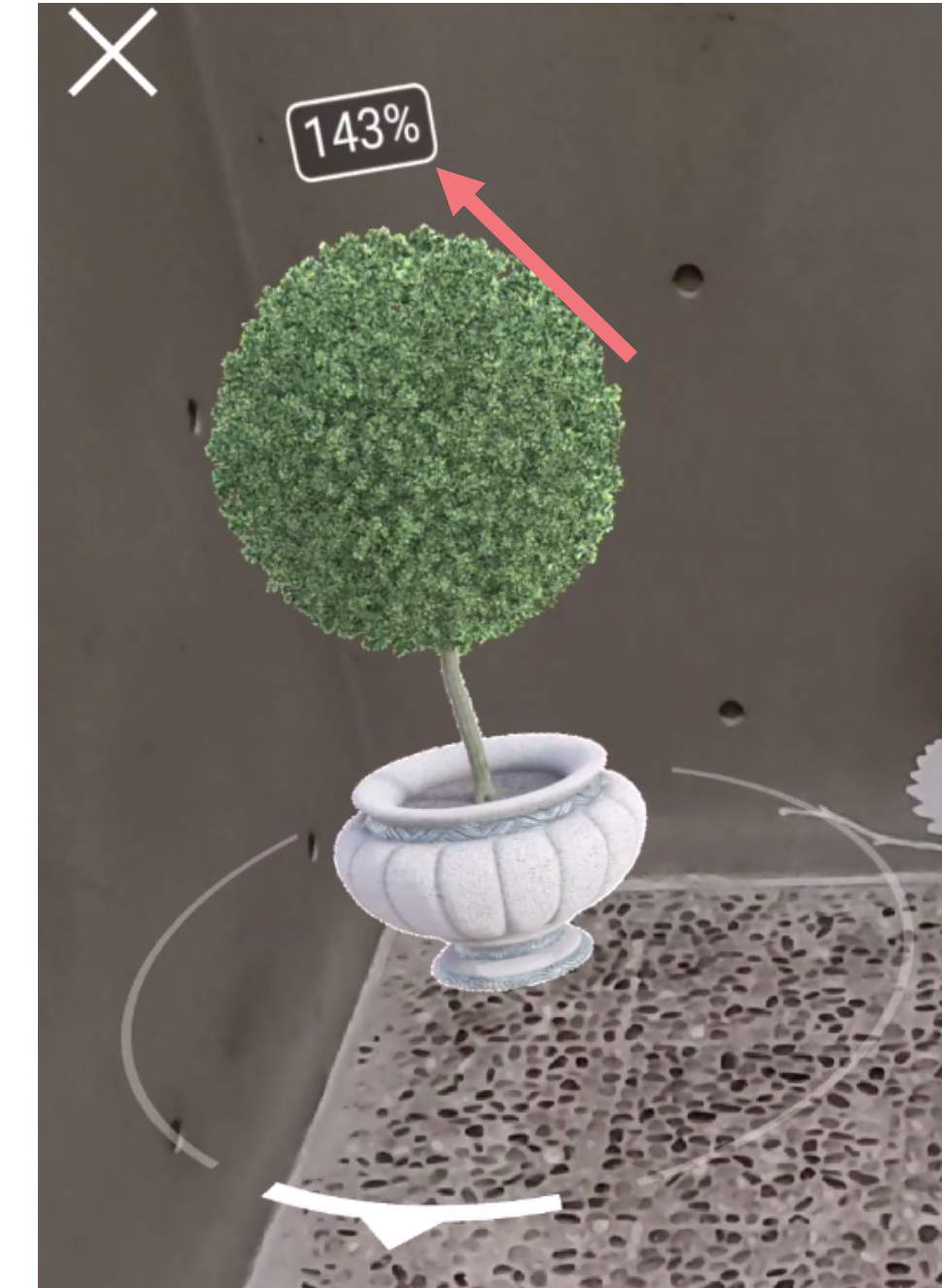
4.2 Control UI



Rotation



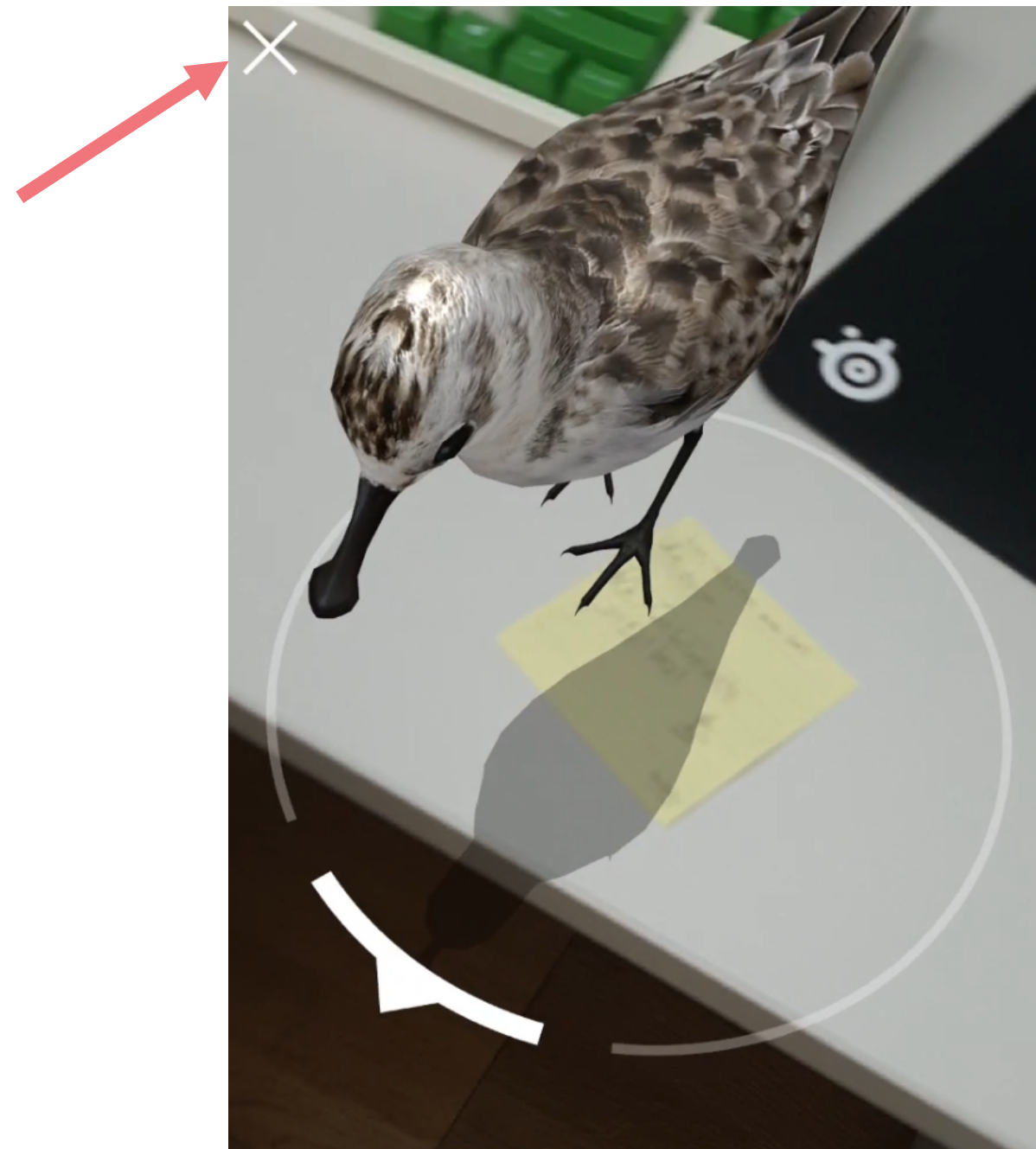
Translation



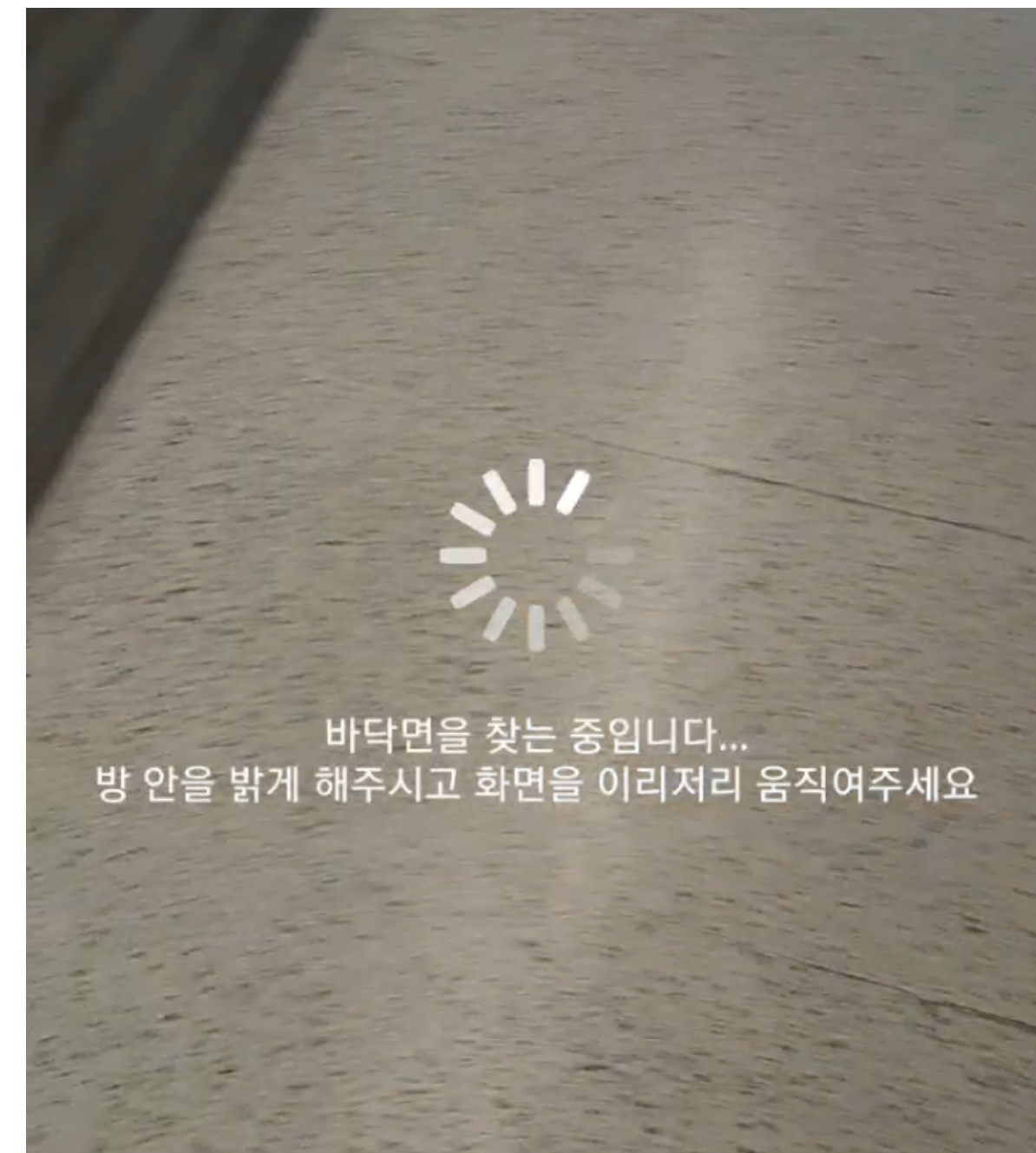
Scale

컨트롤시 반응하는 UI - WebGL

4.3 Overlay UI



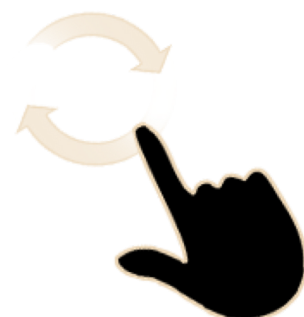
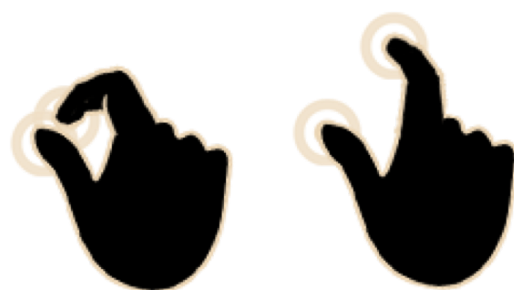
세션 종료 버튼



모델 표시 전 안내문 표시

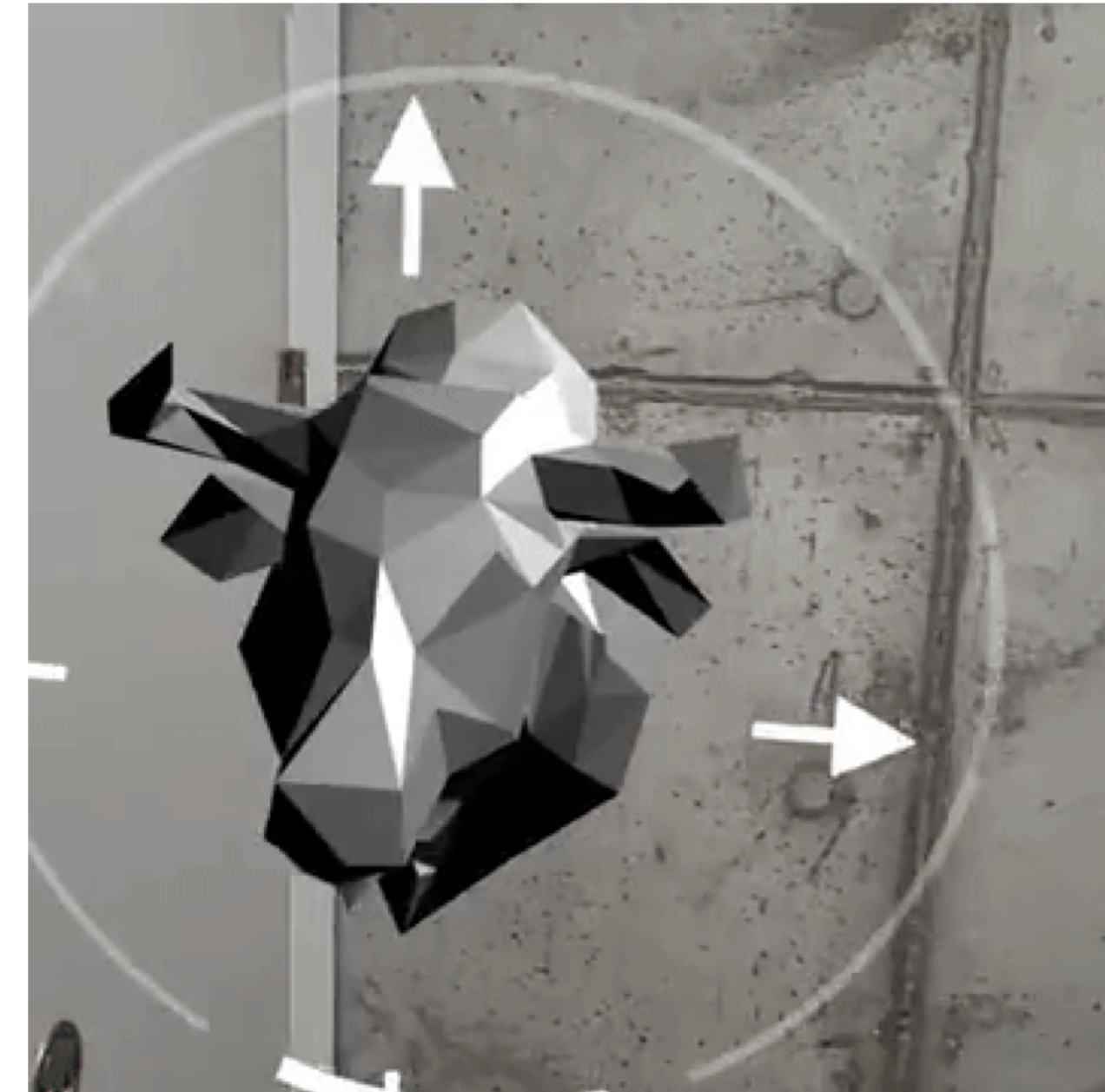
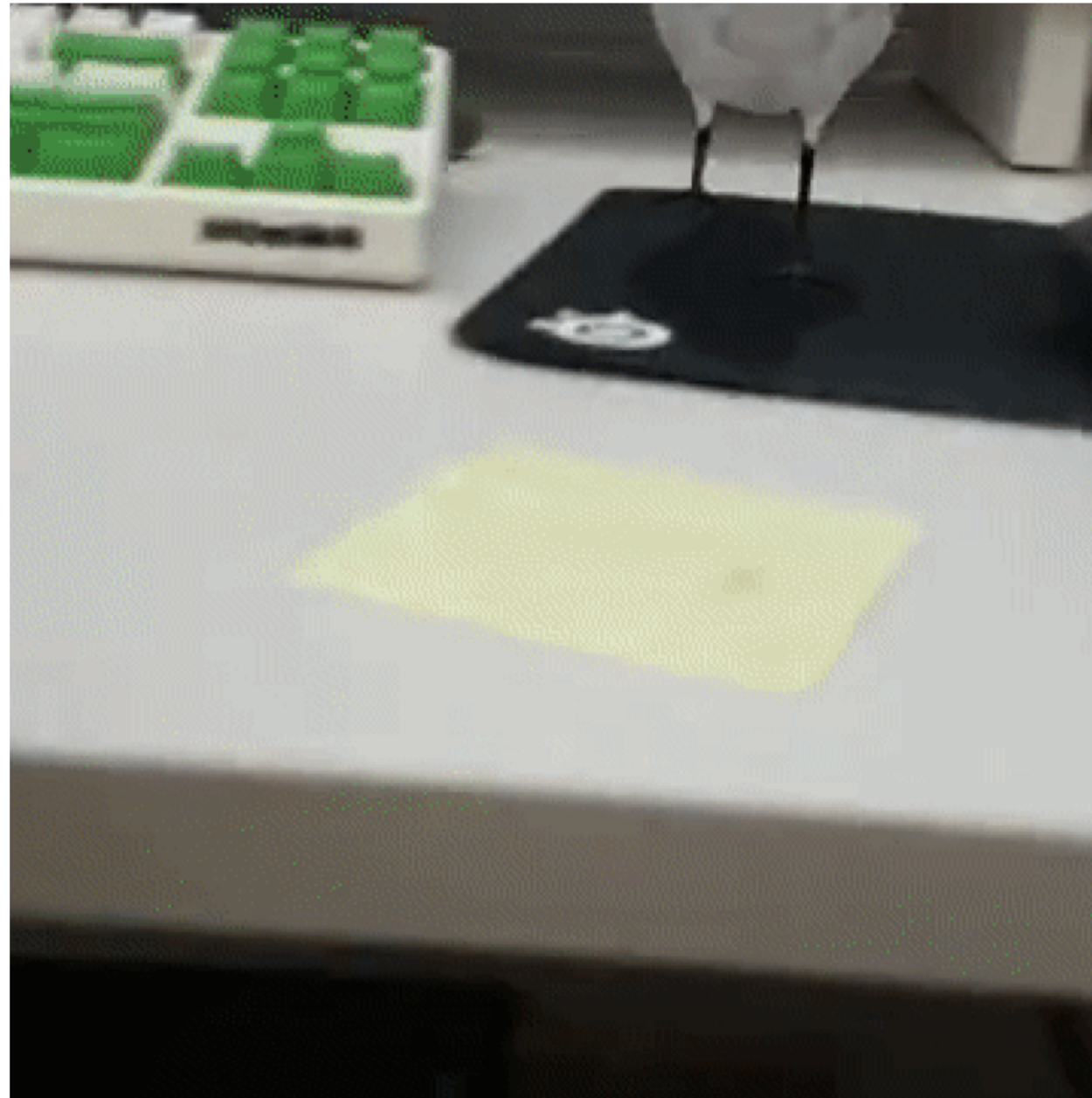
모델 표시 상태 / 인터랙션 가능한 버튼 등
WebXR Feature 중 dom-overlay 이용

4.4 Control

	Rotation	Translation	Scale
제스처	 모델 바깥 클릭 후 회전	 모델 클릭 후 드래그	 핀치 줌

정해진 표준은 없습니다.

4.5 hit-test 팁..



- 여러 개의 Plane이 detection될 수 있음
- 바닥면이 변경되더라도 사용자가 현재 컨트롤중인 drag plane은 유지

4.6 AR 서비스시 고려사항들

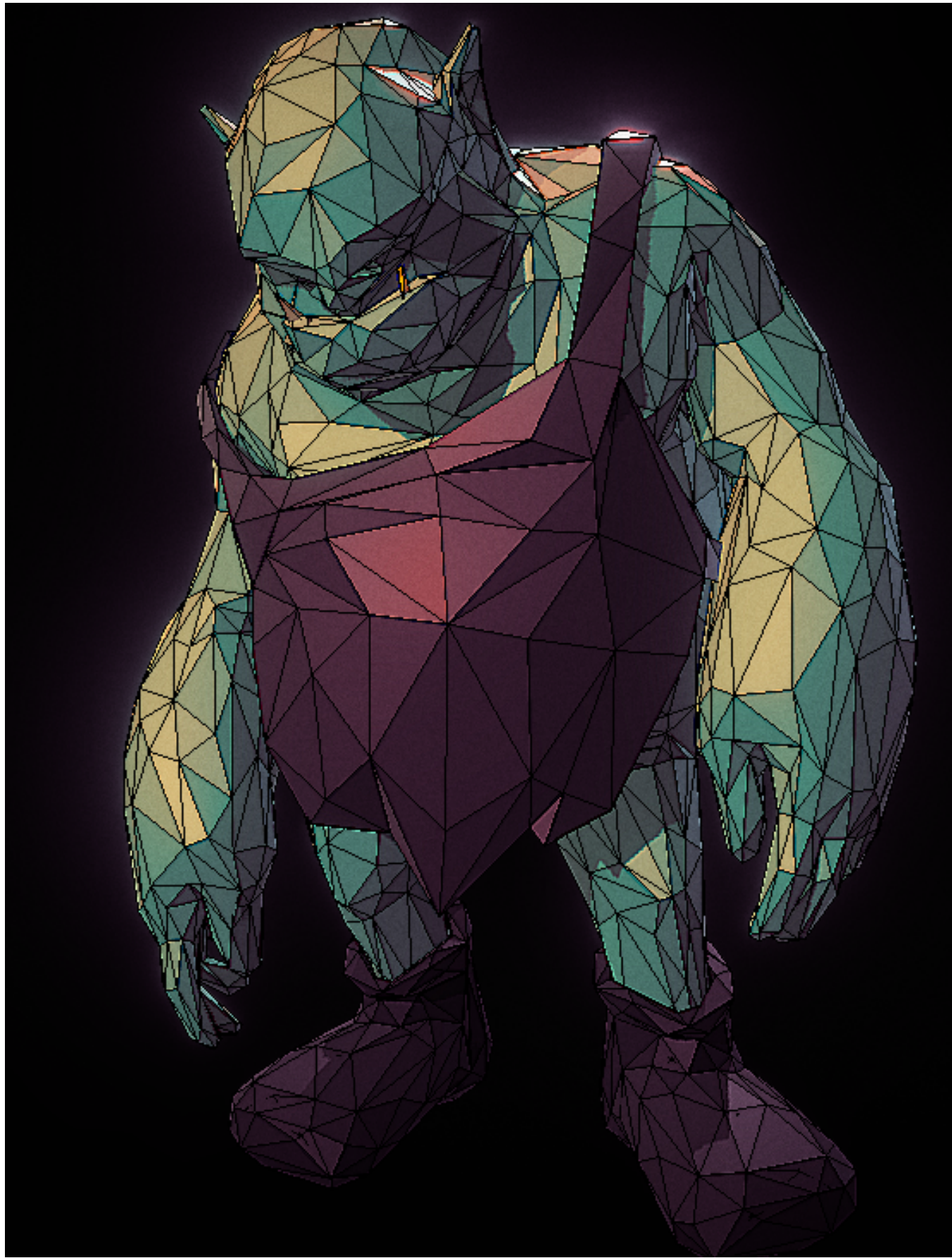


1. 파일 용량 문제
 - 3D 모델의 파일 크기는 다양
 - 복잡하면 복잡할수록 size↑
 - 모델 표시까지 수초~수십초

원본: 55.4MB, gzip: 33.6MB

<https://sketchfab.com/3d-models/dodo-bird-head-sculpt-c4e7f0b9e7974481bb7a385af437d546>

4.6 AR 서비스시 고려사항들



2. Framerate + 발열 & 배터리 소모
- 모델이 복잡할수록 AR 관련 연산량 이외에, 렌더링 비용↑
 - AR에서 Lowpoly 모델을 사용하는 경우가 많음

4.7 File Size Optimization

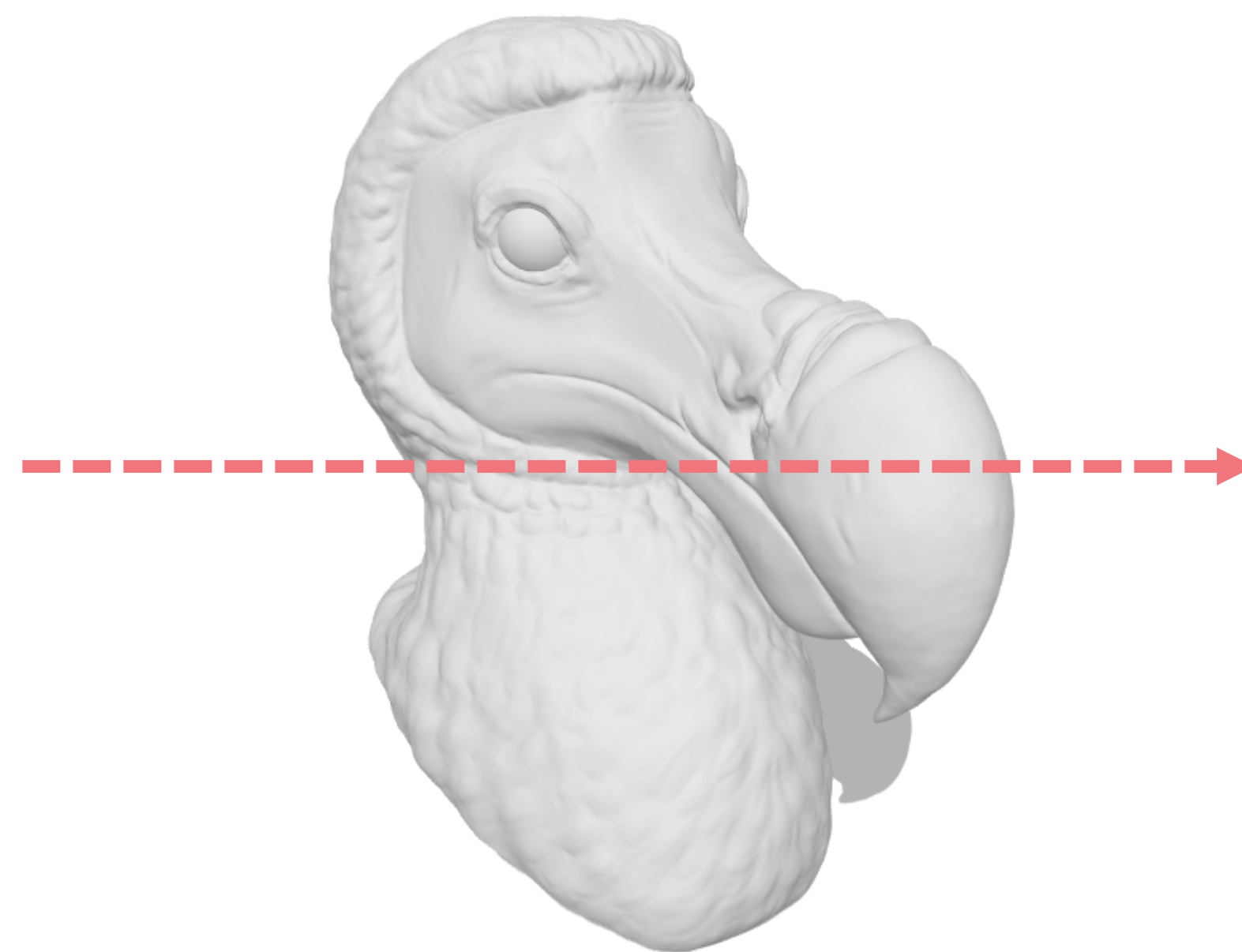
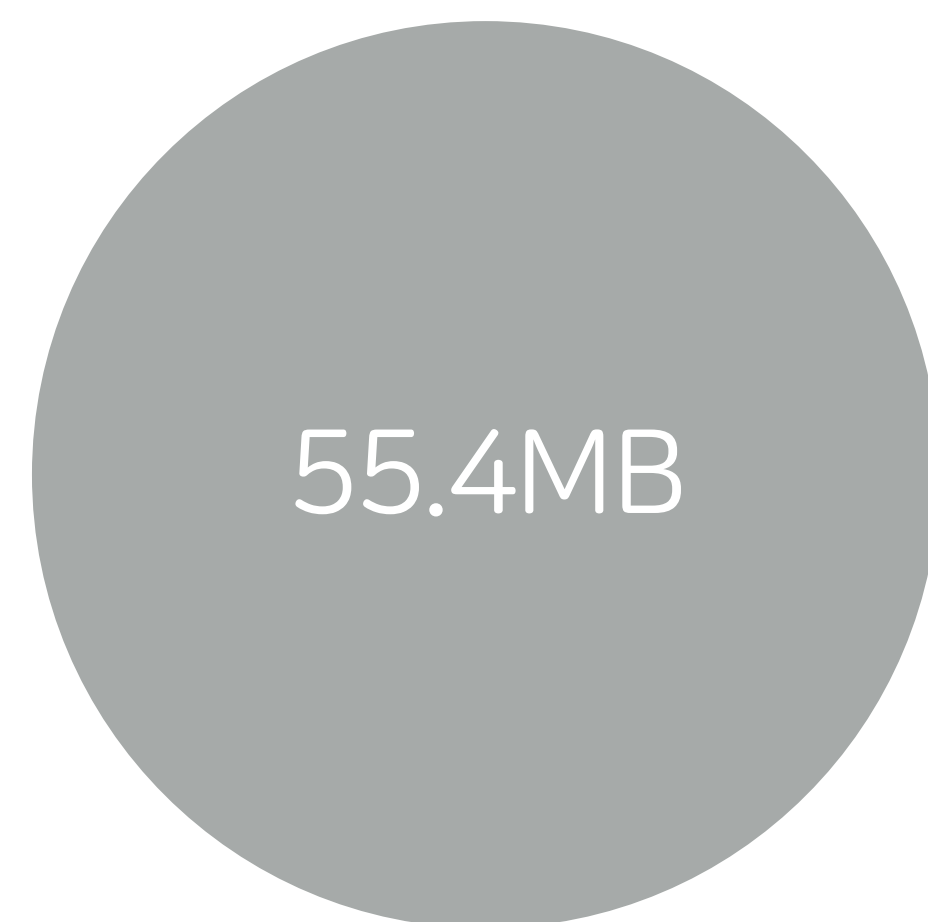
Draco



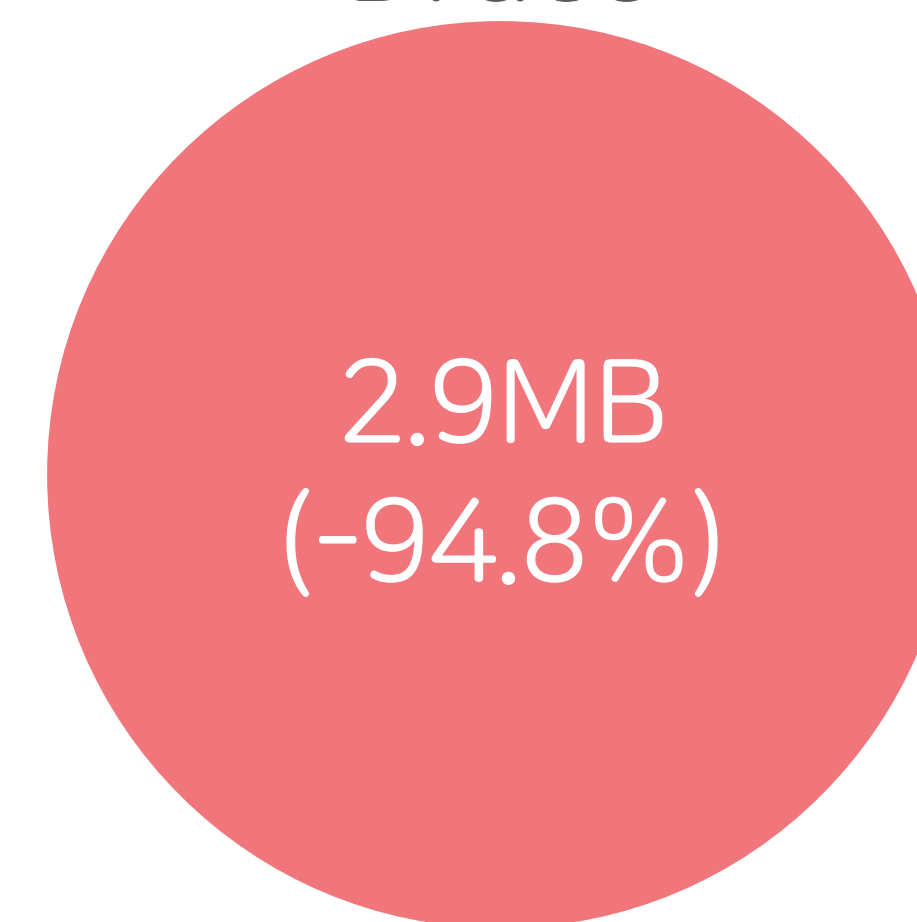
- 구글의 3D 모델 압축 라이브러리
- glTF 포맷에 extension 형태로 적용 가능
- 높은 압축 성능
- 정점 데이터만을 압축

4.7 File Size Optimization

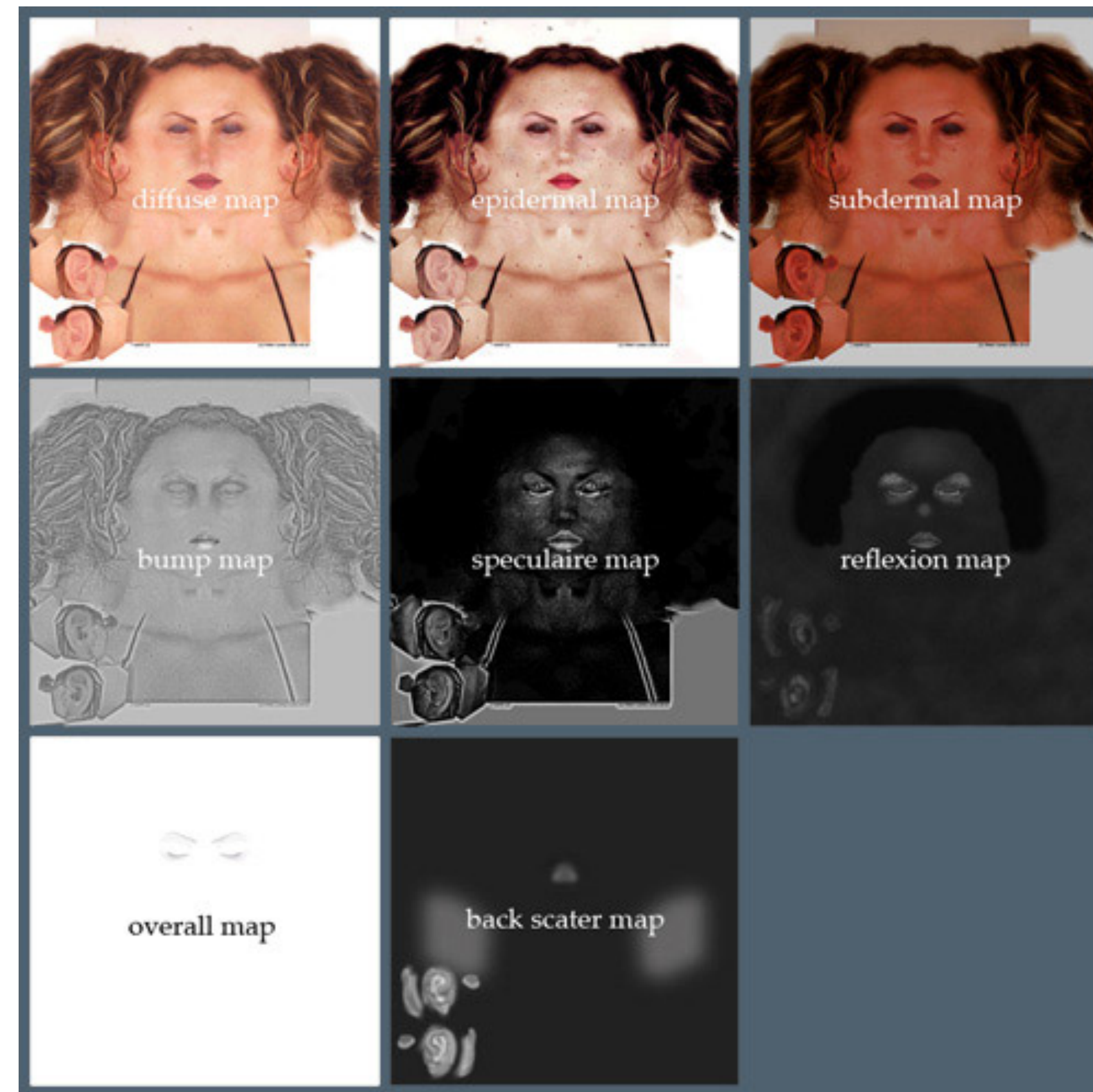
Original glTF size



Draco



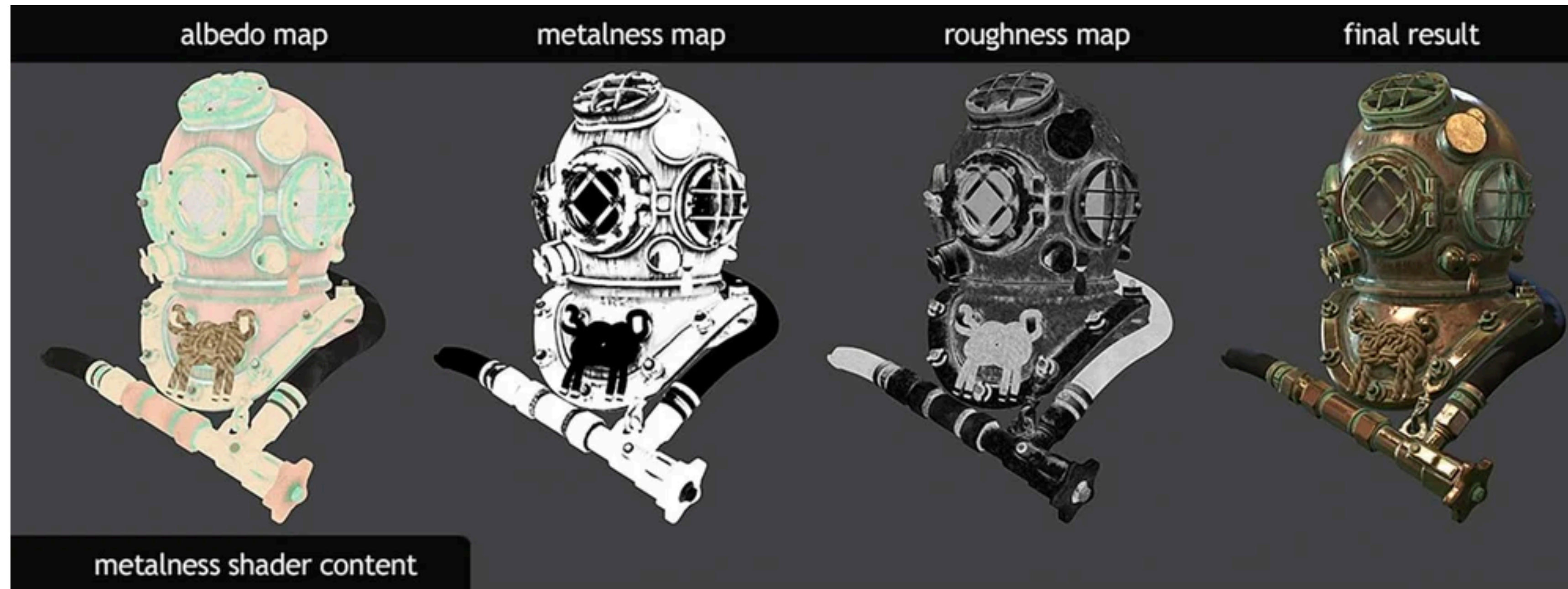
4.7 File Size Optimization



<http://www.cgarena.com/freestuff/tutorials/xsi/realistichead/index2.html>

3D Model - textures

4.7 File Size Optimization



Source: marmoset.co

3D Model – Textures

4.7 File Size Optimization



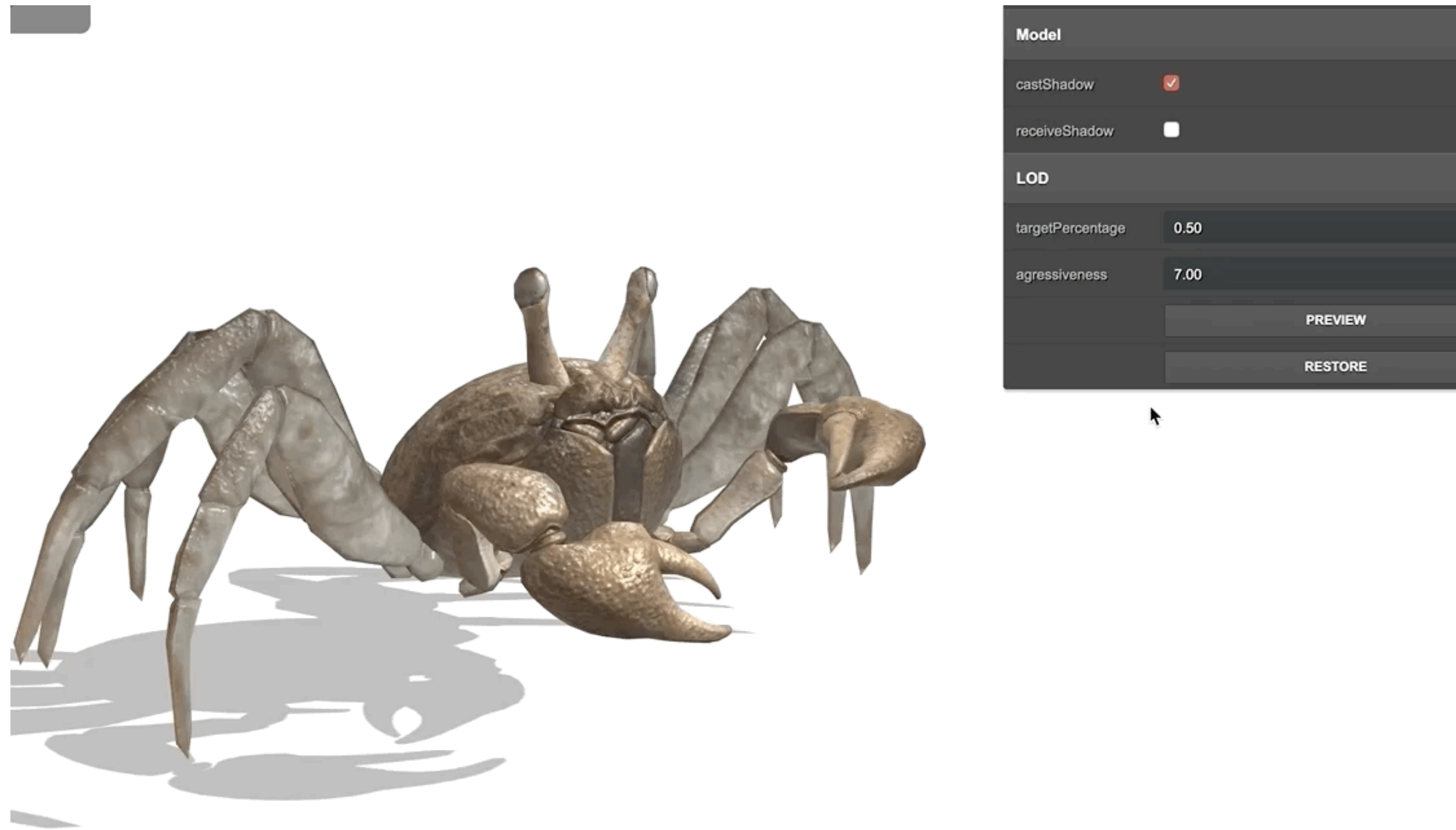
Frescos at Medieval Convent, São Cucufate, Port., Sketchfab

glTF size: 44.3MB

Texture size: 221.5MB

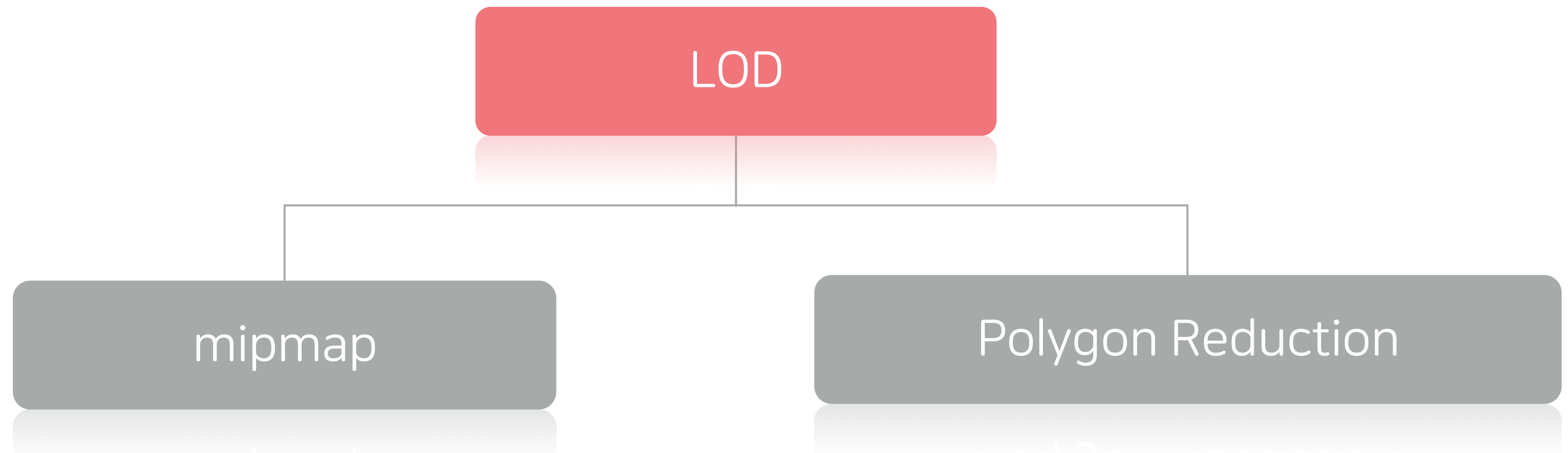
-> 최대로 압축하더라도 221.5MB

4.8 File Size Optimization – LOD



LOD(Level of Detail)

4.8 File Size Optimization – LOD



4.8 File Size Optimization – LOD



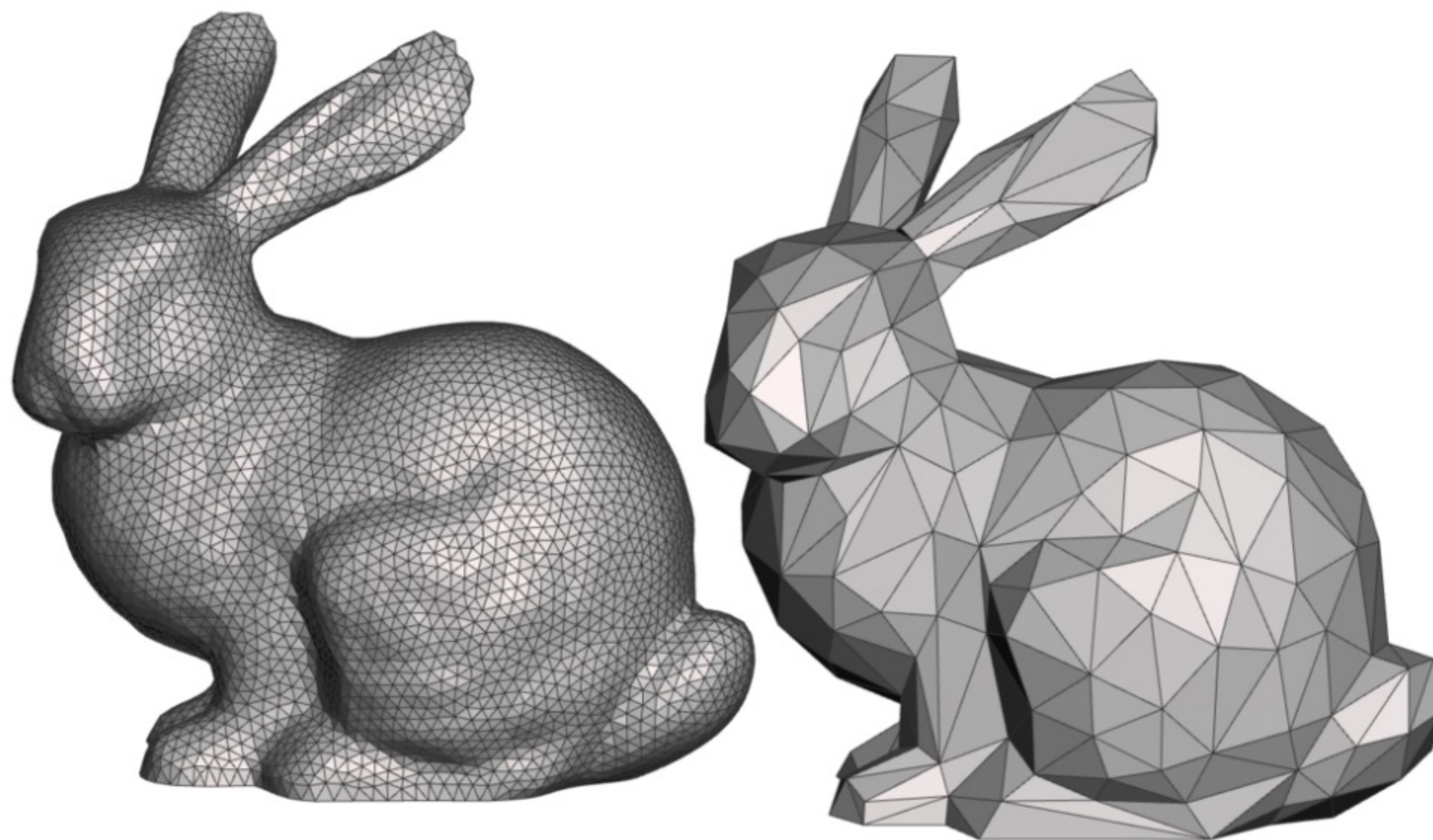
<https://www.tomshardware.com/reviews/atj,819-2.html>

mipmap

4.8 File Size Optimization – LOD

원본 텍스처	크기 축소된 텍스처
	
1024*1024	128*128
521KB	12KB(-97.7%)

4.8 File Size Optimization – LOD



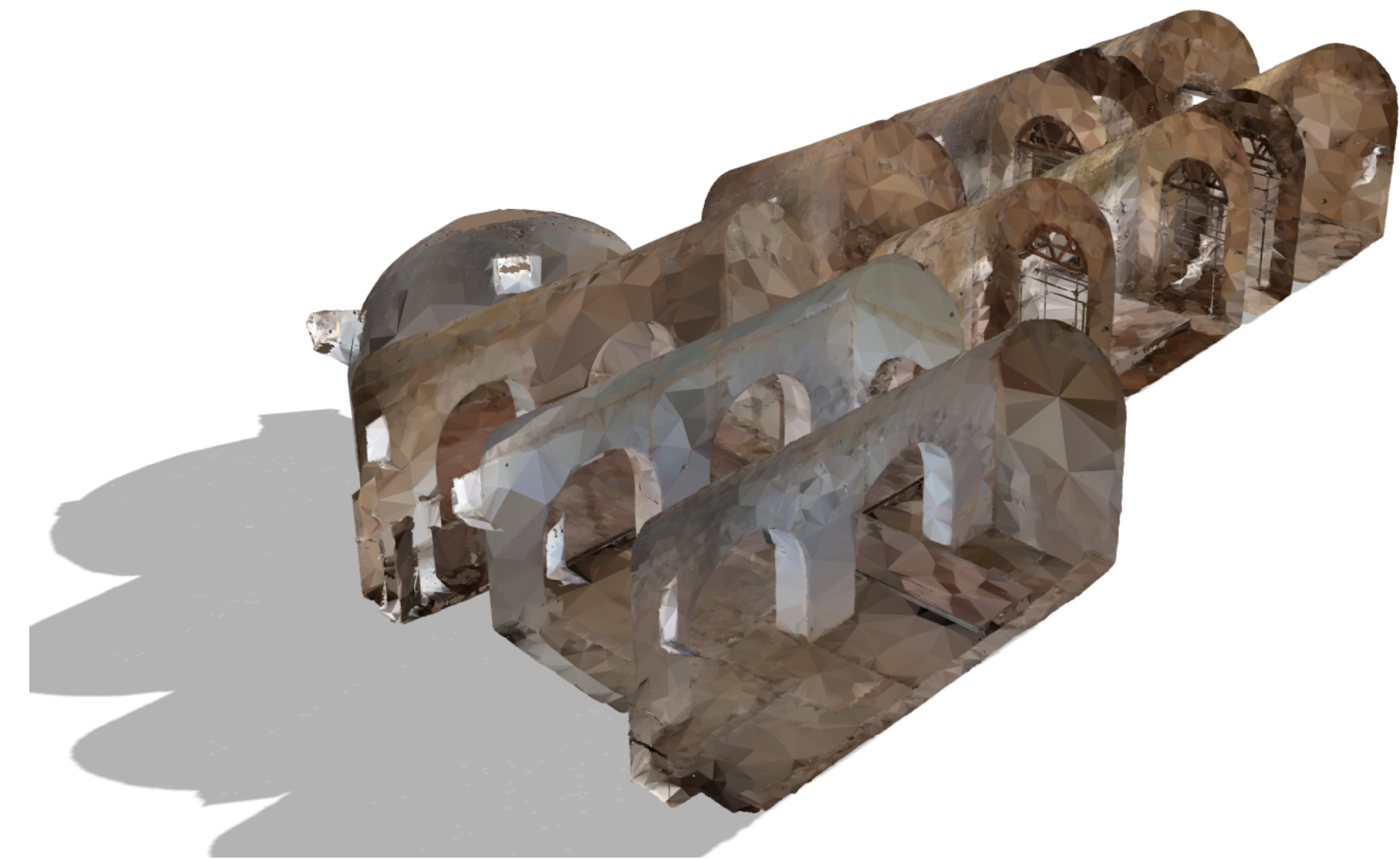
<https://www.tomshardware.com/reviews/atj,819-2.html>

Polygon reduction
(mesh-simplifier)

4.8 File Size Optimization – LOD

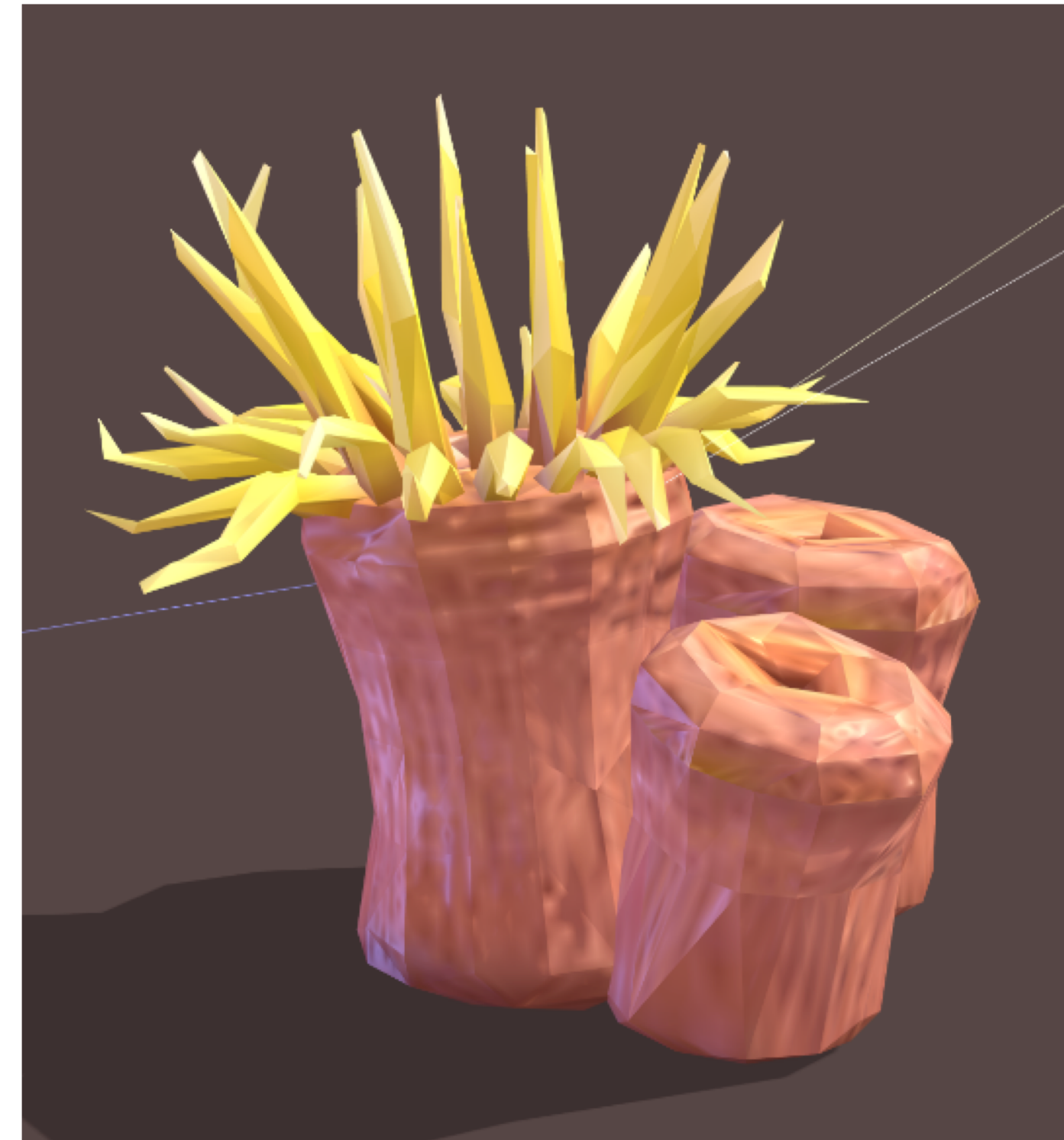
	원본 모델	간소화된 모델
		
면의 개수	1486290	176948(-88.1%)
점의 개수	809995	155322(-80.8%)
크기	55.4MB	12.8MB
Draco 적용 후 크기	2.9MB	1.1MB

4.8 File Size Optimization – LOD



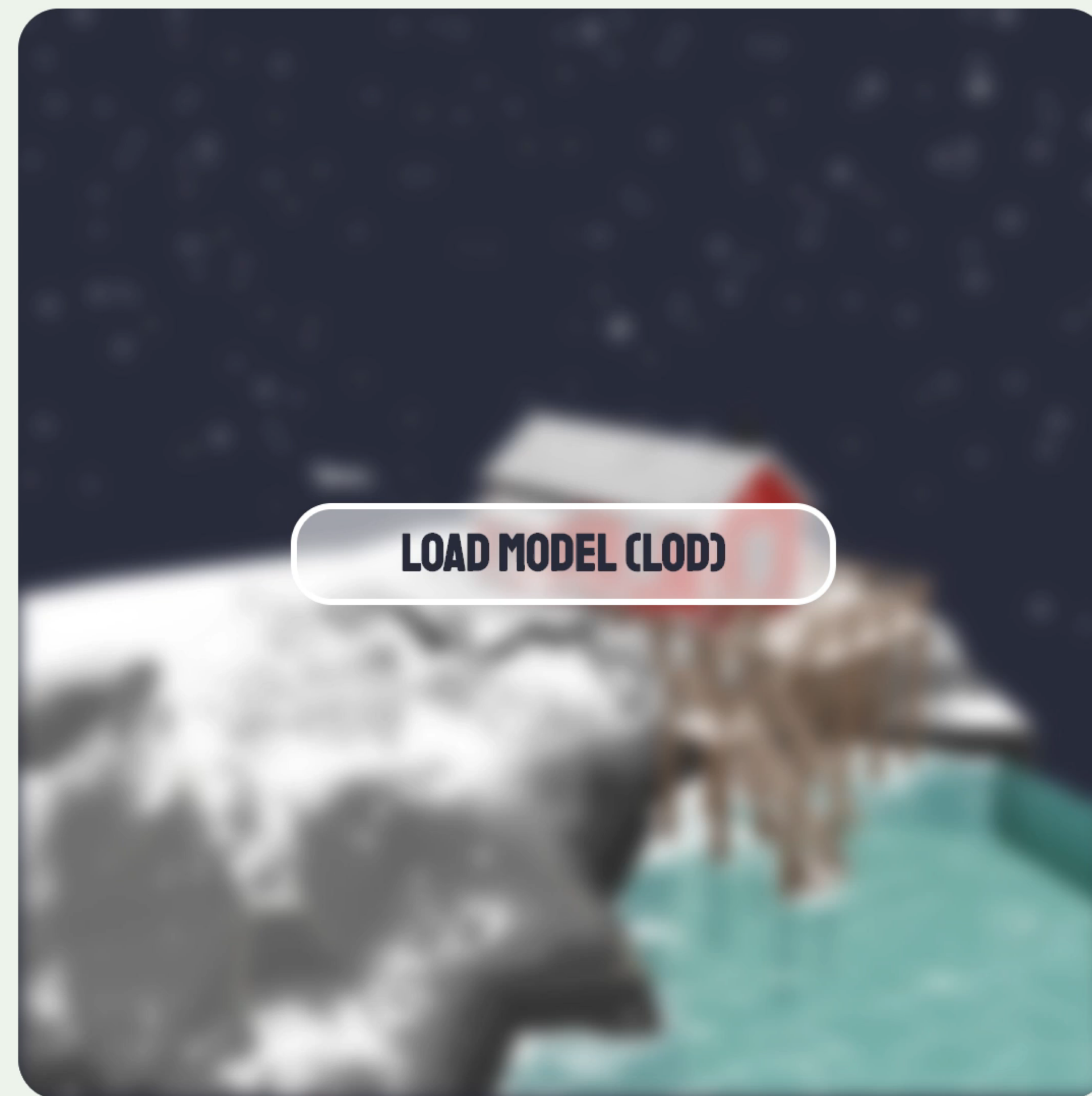
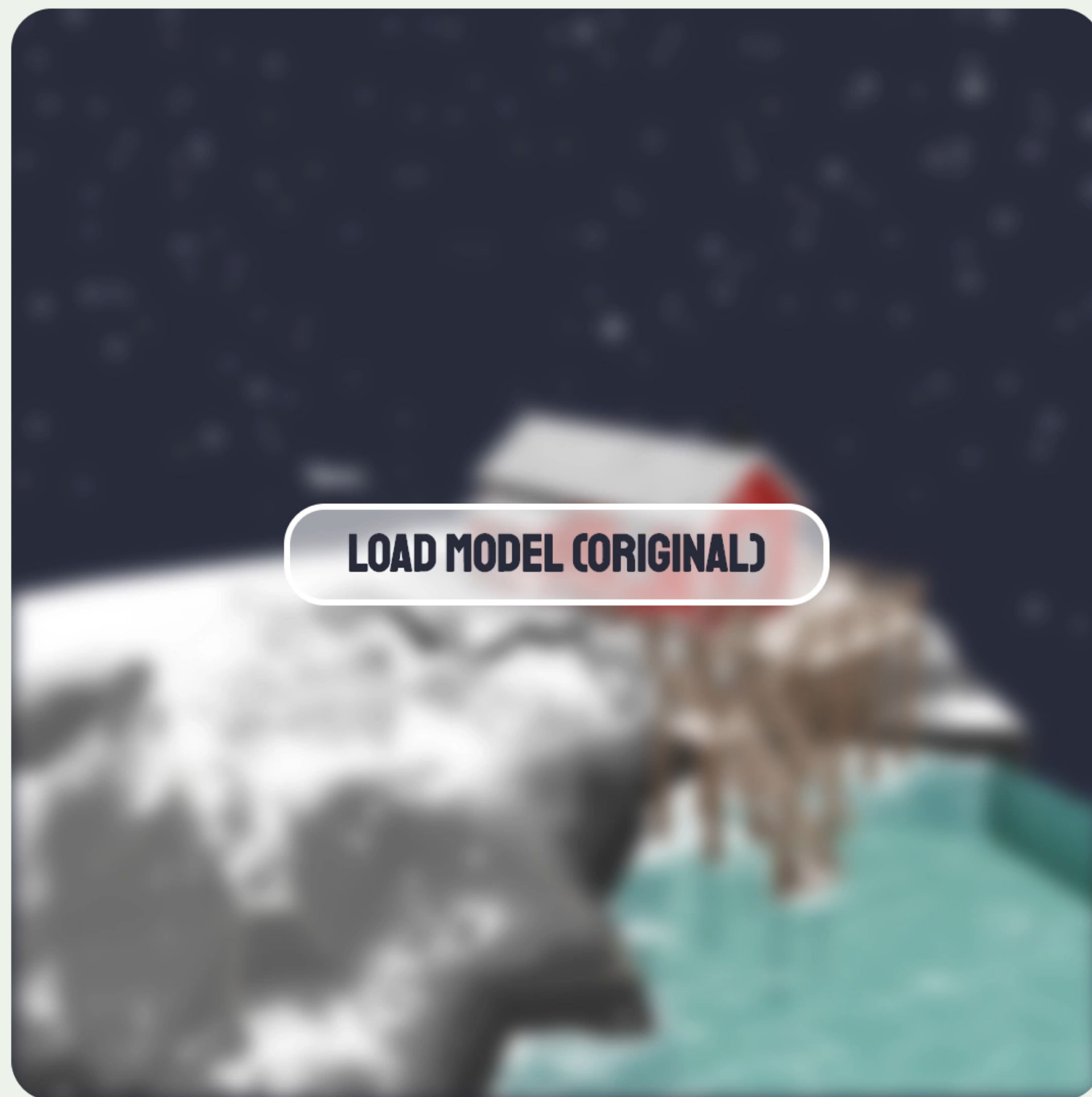
265.8MB -> 4.2MB(-98.28%)

4.8 File Size Optimization – LOD



4.3MB -> 47KB (-98.9%)

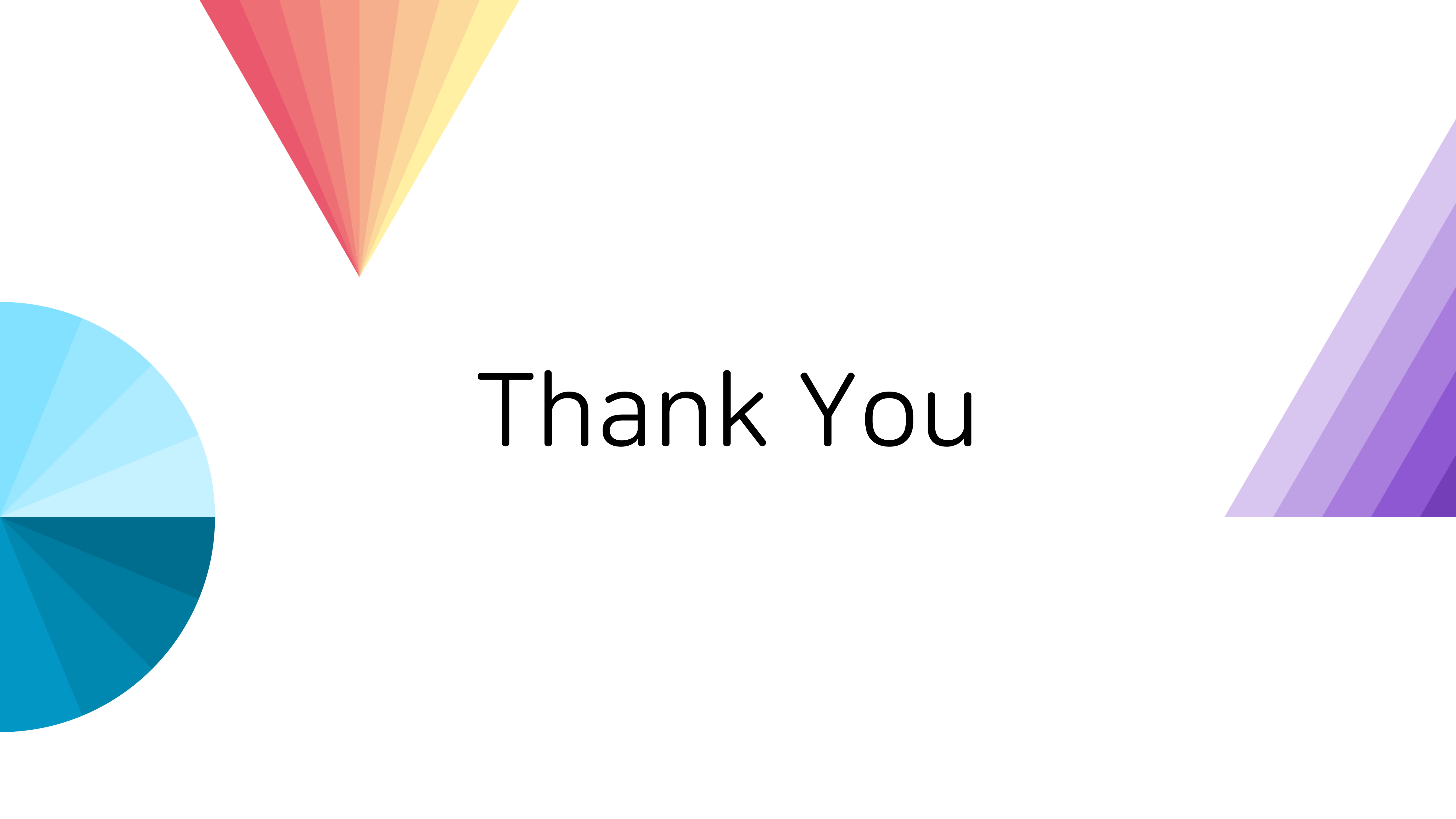
4.8 File Size Optimization – LOD



4.8 File Size Optimization – LOD

Draco – 주의점

- Draco의 경우 decoder가 자체적인 용량을 갖고 있음
- 파일 용량 감소에 의한 다운로드 시간 이득에 비해 Decoder에 의한 디코딩 시간이 더 오래 걸릴 경우 큰 의미가 없음



Thank You



Q & A