Stony Brook University

Erebus: Access Control for Augmented Reality Systems

Sanket Goutam*, Yoonsang Kim*, Amir Rahmati, Arie Kaufman

Stony Brook University

** equal contribution*

Two form factors for building AR Systems

Standalone

Oculus Quest 2

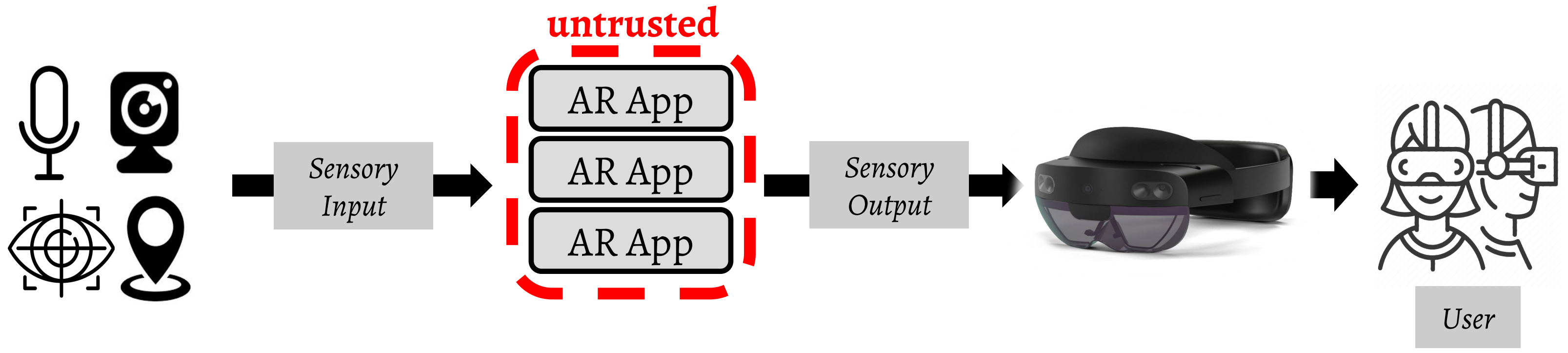
HoloLens 2

Companion Device

Toshiba dynaEdge

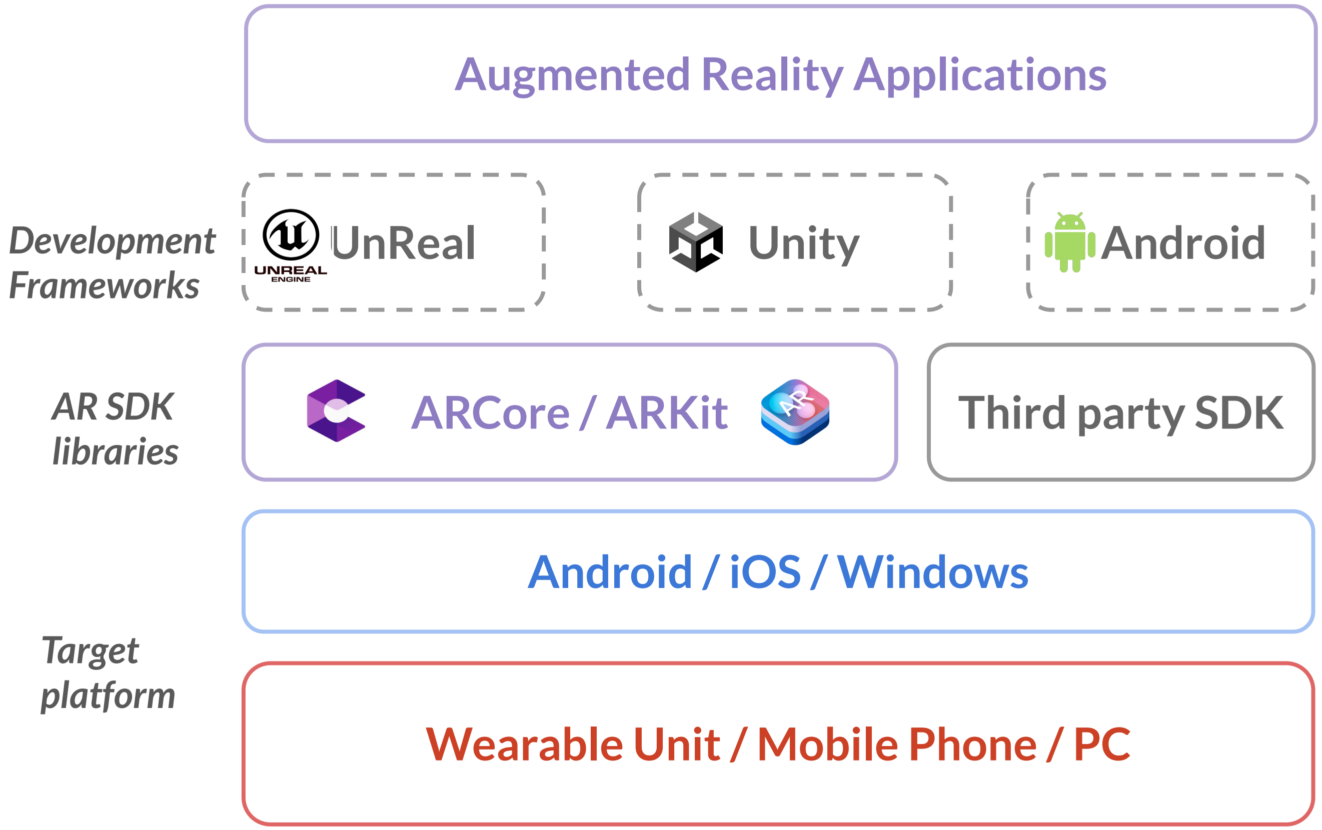
Rokid Air Pro

Applications derive information from device sensors.

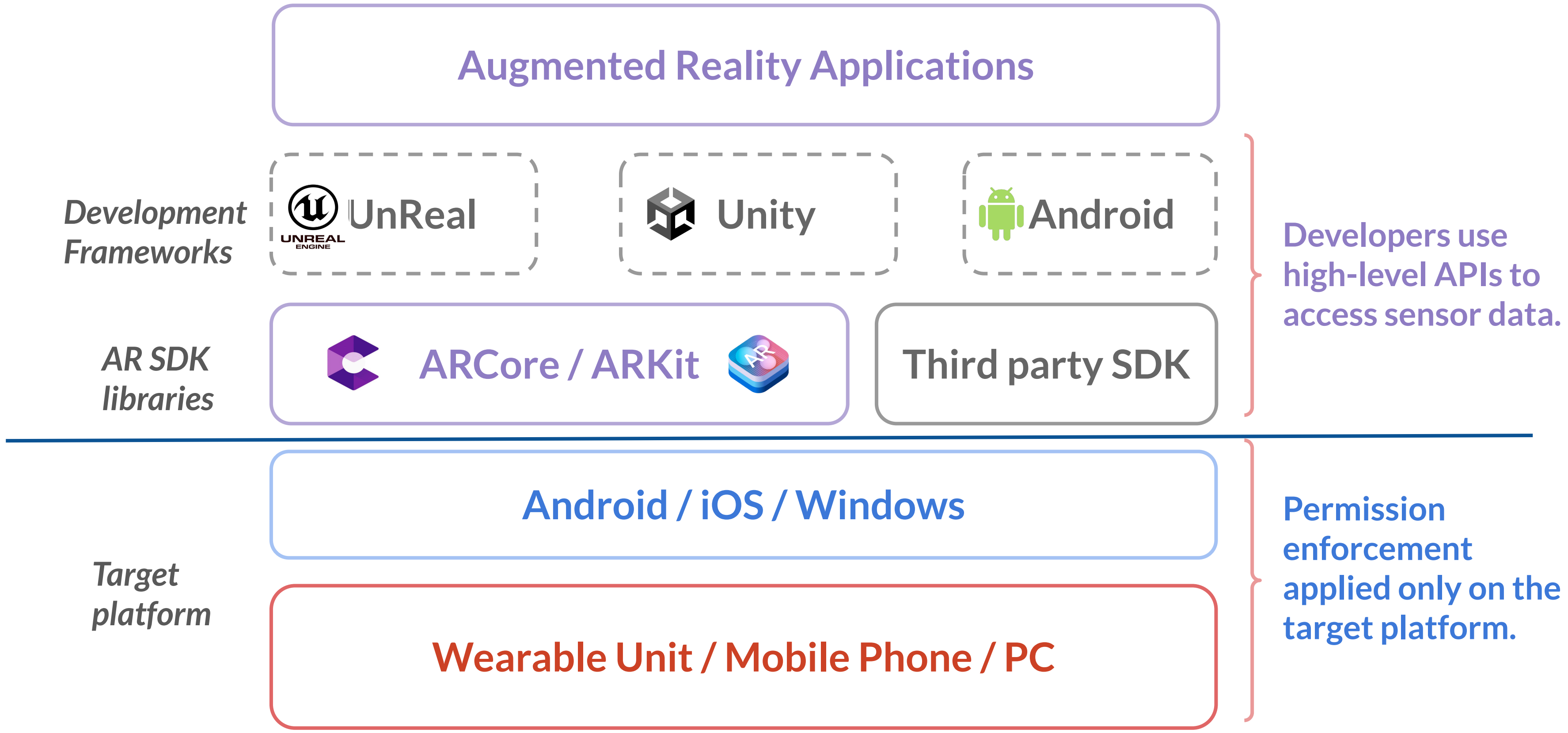


Application Pipeline

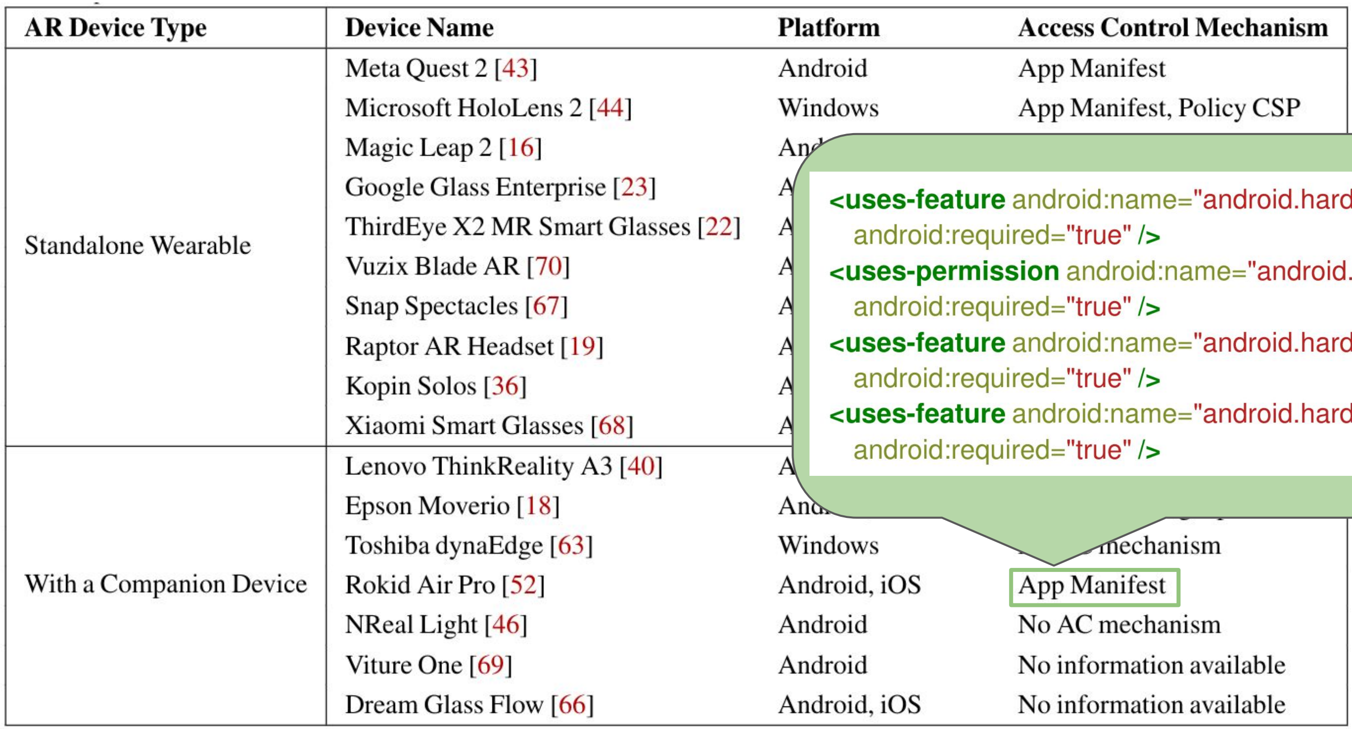
How are these applications developed?



Dichotomy between data required and access requested.


Augmented Reality Applications
Development Frameworks
UnReal
UNREAL ENGINE
Unity
Android
AR SDK libraries
ARCore / ARKit
Third party SDK
Developers use high-level APIs to access sensor data.
Target platform
Android / iOS / Windows
Wearable Unit / Mobile Phone / PC
Permission enforcement applied only on the target platform.

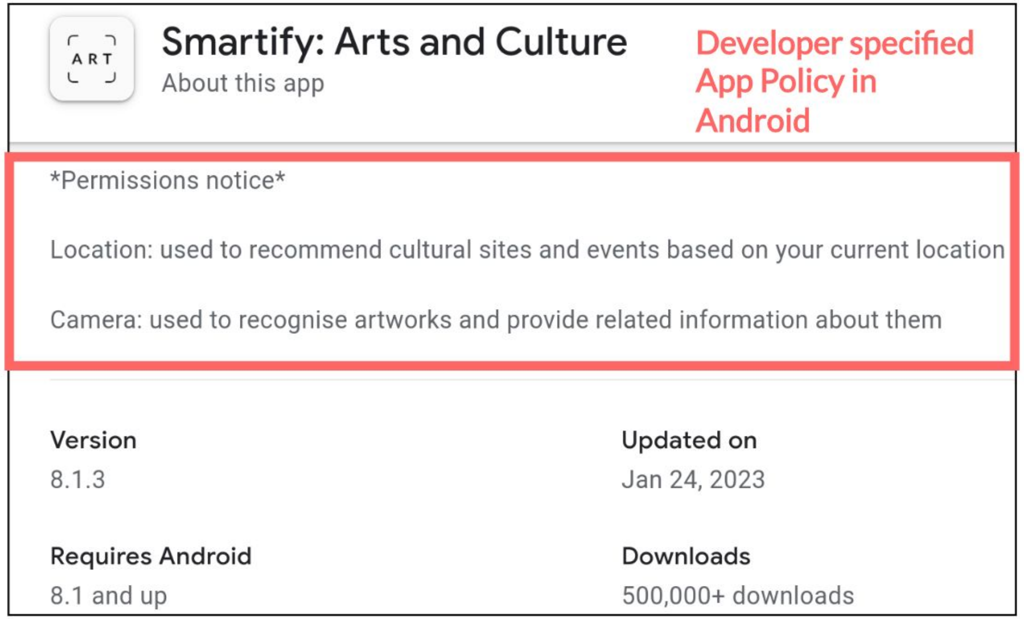
Permission Control similar to Smartphone OS.



AR Device Type	Device Name	Platform	Access Control Mechanism
Standalone Wearable	Meta Quest 2 [43]	Android	App Manifest
	Microsoft HoloLens 2 [44]	Windows	App Manifest, Policy CSP
	Magic Leap 2 [16]	And	
	Google Glass Enterprise [23]	A	
	ThirdEye X2 MR Smart Glasses [22]	A	
	Vuzix Blade AR [70]	A	
	Snap Spectacles [67]	A	
	Raptor AR Headset [19]	A	
	Kopin Solos [36]	A	
	Xiaomi Smart Glasses [68]	A	
With a Companion Device	Lenovo ThinkReality A3 [40]	A	
	Epson Moverio [18]	And	
	Toshiba dynaEdge [63]	Windows	mechanism
	Rokid Air Pro [52]	Android, iOS	App Manifest
	NReal Light [46]	Android	No AC mechanism
	Viture One [69]	Android	No information available
	Dream Glass Flow [66]	Android, iOS	No information available

```
<uses-feature android:name="android.hardware.camera"
  android:required="true" />
<uses-permission android:name="android.permission.record_audio"
  android:required="true" />
<uses-feature android:name="android.hardware.location.GPS"
  android:required="true" />
<uses-feature android:name="android.hardware.sensor.heartrate"
  android:required="true" />
```

Developer specifies an access policy to user on Play Store.



Smartify: Arts and Culture

About this app

Developer specified App Policy in Android

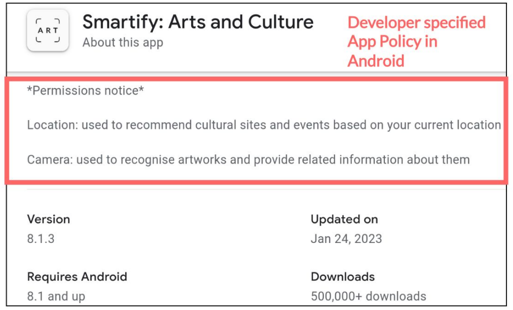
Permissions notice

Location: used to recommend cultural sites and events based on your current location

Camera: used to recognise artworks and provide related information about them

Version	Updated on
8.1.3	Jan 24, 2023
Requires Android	Downloads
8.1 and up	500,000+ downloads

User installs the app on their device.



Smartify: Arts and Culture
About this app

Developer specified App Policy in Android

Permissions notice

Location: used to recommend cultural sites and events based on your current location

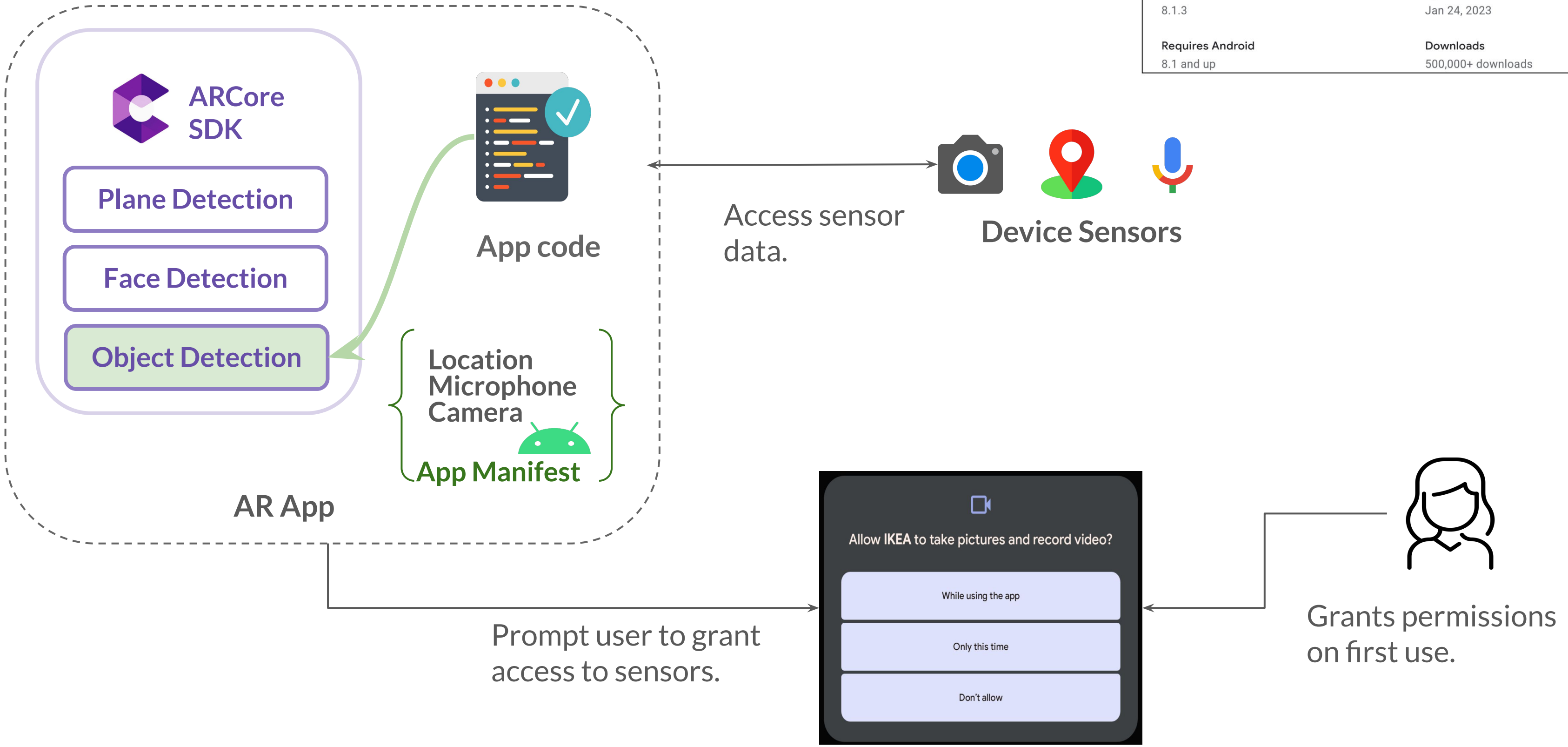
Camera: used to recognise artworks and provide related information about them

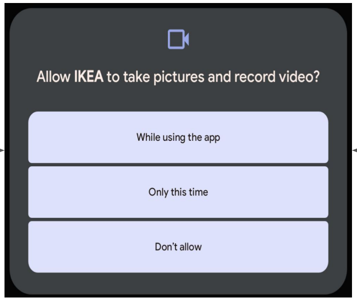
Version
8.1.3

Updated on
Jan 24, 2023

Requires Android
8.1 and up

Downloads
500,000+ downloads





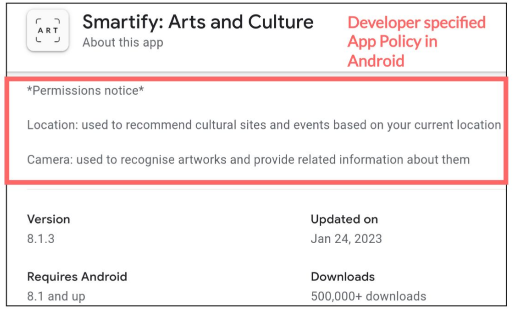
Allow IKEA to take pictures and record video?

While using the app

Only this time

Don't allow

Malicious app can violate app policy.


Smartify: Arts and Culture
About this app

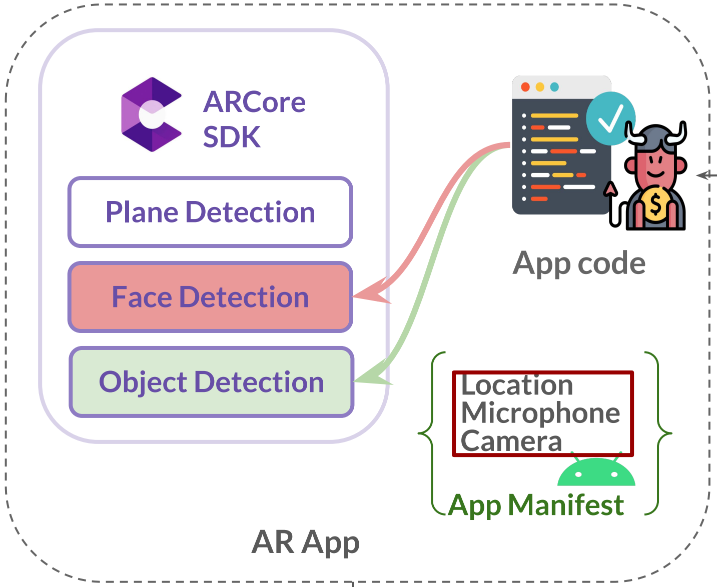
Developer specified App Policy in Android

Permissions notice

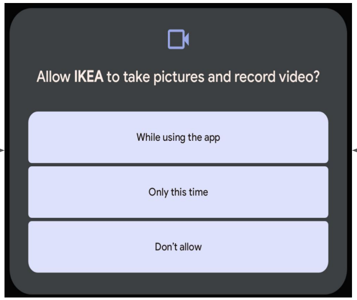
Location: used to recommend cultural sites and events based on your current location

Camera: used to recognise artworks and provide related information about them

Version	Updated on
8.1.3	Jan 24, 2023
Requires Android	**Downloads**
8.1 and up	500,000+ downloads



Access sensor data.

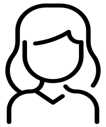

Allow IKEA to take pictures and record video?

While using the app

Only this time

Don't allow

Prompt user to grant access to sensors.



Grants permissions on first use.

Can we reimagine Access Control for VisionOS?

SPATIAL COMPUTING —

Unity's visionOS support has started to roll out—here's how it works

A closed beta will admit developers gradually over the coming weeks.

SAMUEL AXON - 7/19/2023, 3:51 PM

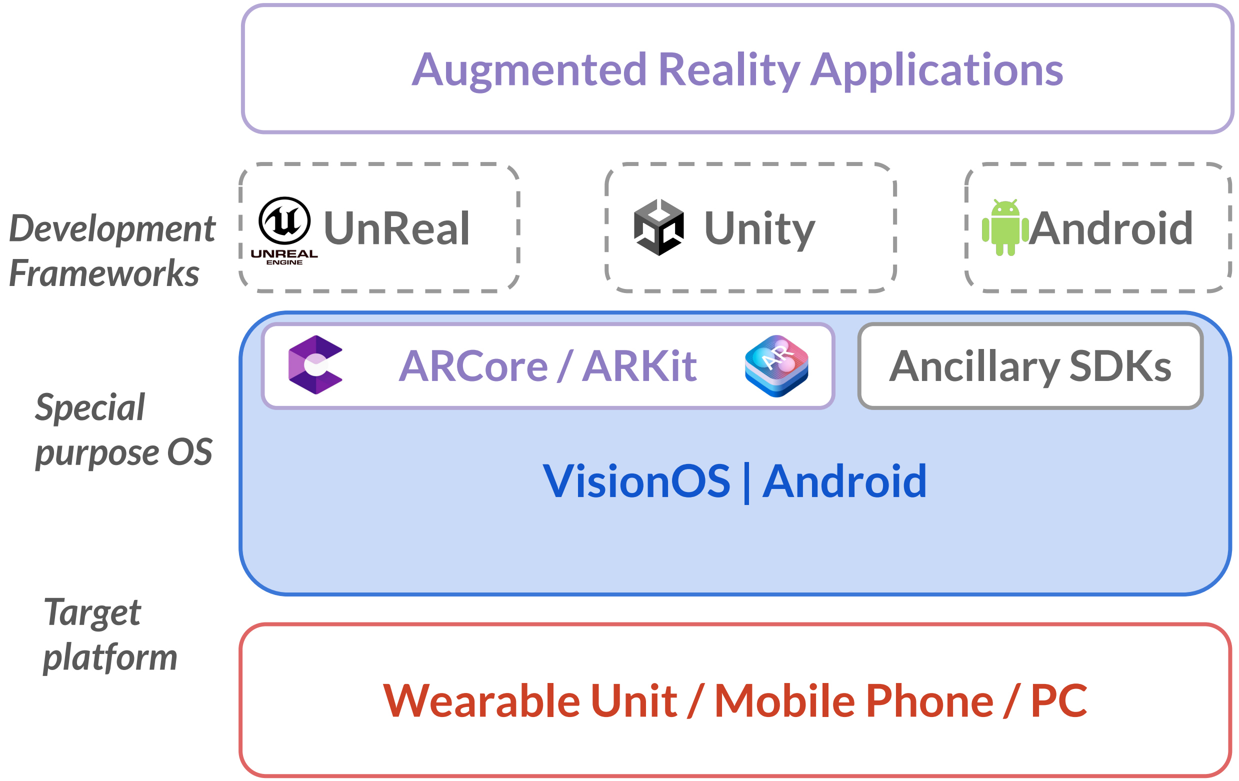
Discover visionOS
All-new platform. Familiar frameworks and tools. Get ready to design and build an entirely new universe of apps and games for Apple Vision Pro.

G1. How can we regulate direct access to sensors?

G2. How to ensure a least privilege access based on developer-specified policy, allowing access to what's required and nothing more?

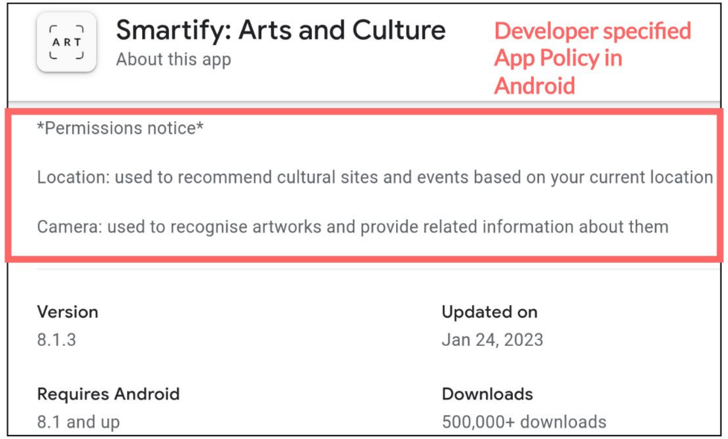
G3. Can we allow users to adjust access based on their requirements?

Erebus: regulating sensor access at the OS-level



Integrating AR functional requirements with permission control at the OS-level.

Erebus: policy specification language that *expresses* functionality


Smartify: Arts and Culture
About this app

Developer specified App Policy in Android

Permissions notice

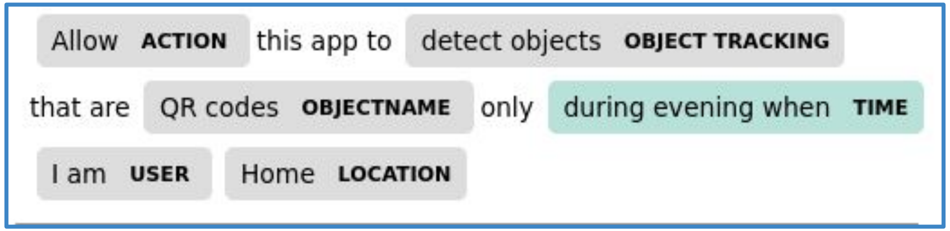
Location: used to recommend cultural sites and events based on your current location

Camera: used to recognise artworks and provide related information about them

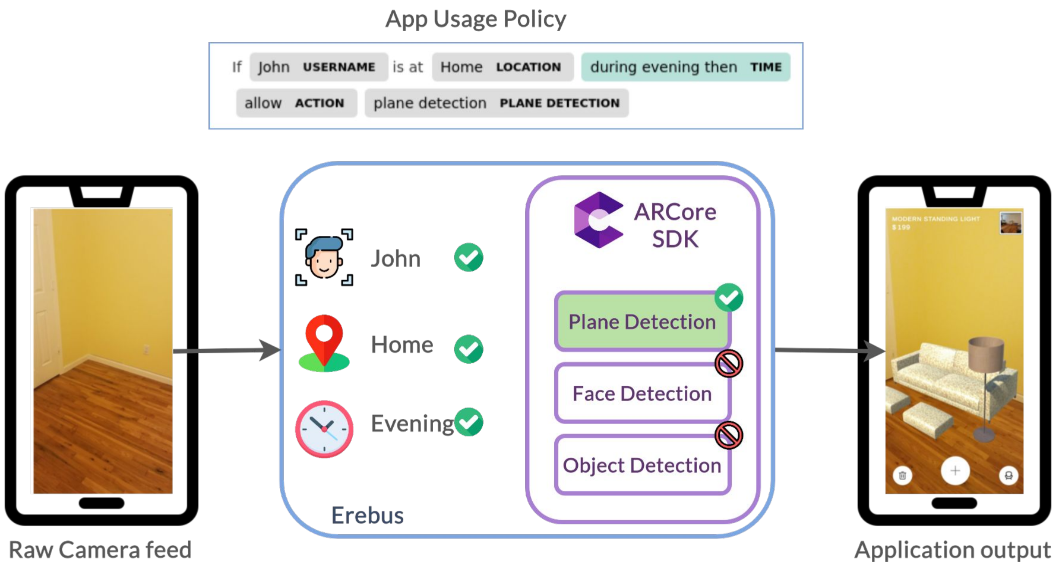
Version	Updated on
8.1.3	Jan 24, 2023
Requires Android	**Downloads**
8.1 and up	500,000+ downloads

- Coarse-grained access requirement. **(Location, Camera)**
- Functional requirement cannot be enforced by the system **(recognize artworks).**

- Functional description in a semi-structured natural language format.
- Fine-grained permission enforcement.


Allow ACTION this app to detect objects OBJECT TRACKING that are QR codes OBJECTNAME only during evening when TIME I am USER Home LOCATION

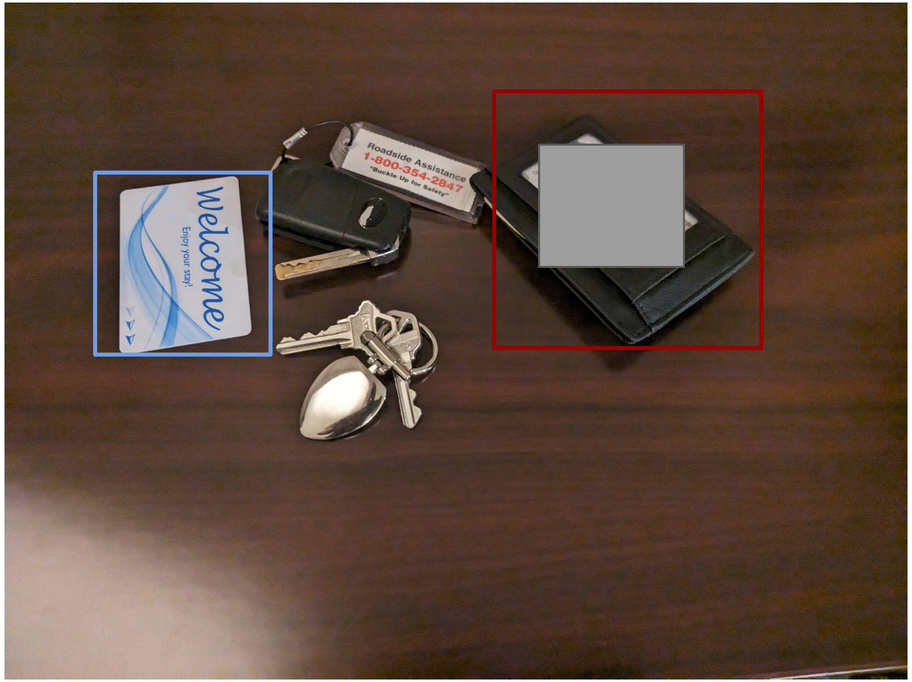
Erebus: users define *what*, *when*, *and where* data can be accessed



System validates app's sensor request based on context-dependent policy specification, ensuring ***least-privilege.***

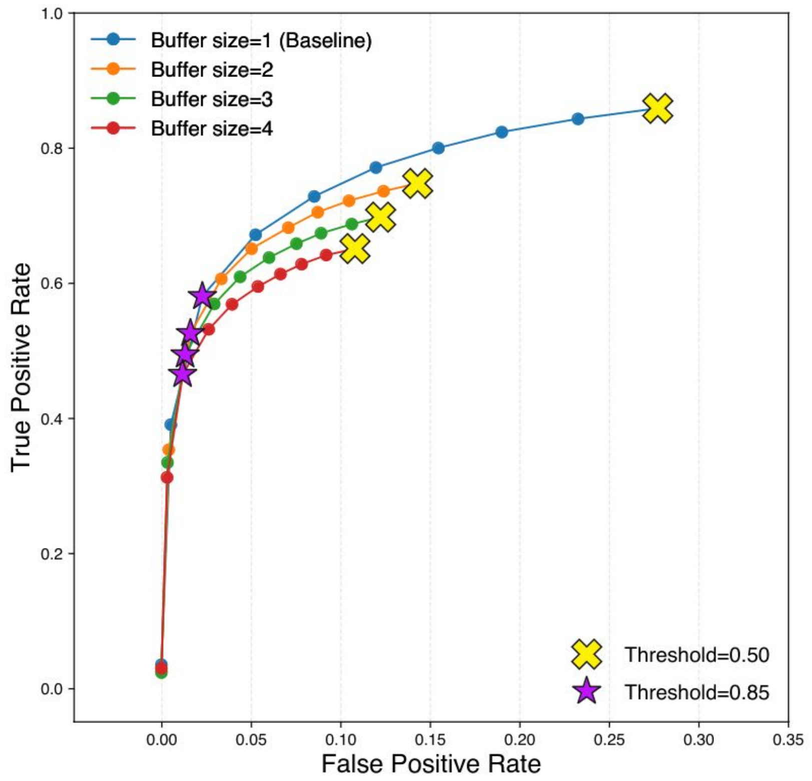
Erebus: preventing sensitive data leakage

Object detection is an imperfect process. False positives could leak sensitive information to the app.

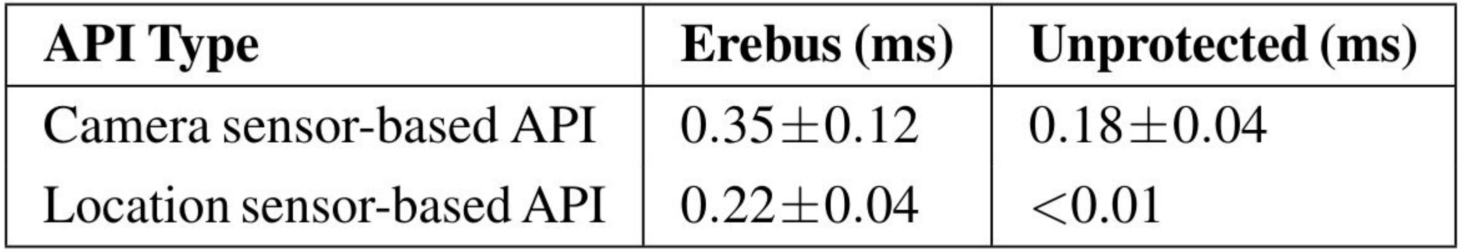


How does Erebus prevent leaking sensitive data due to false positives?

We leverage *conflation* technique to optimize object detection accuracy and reduce false positives in Erebus.



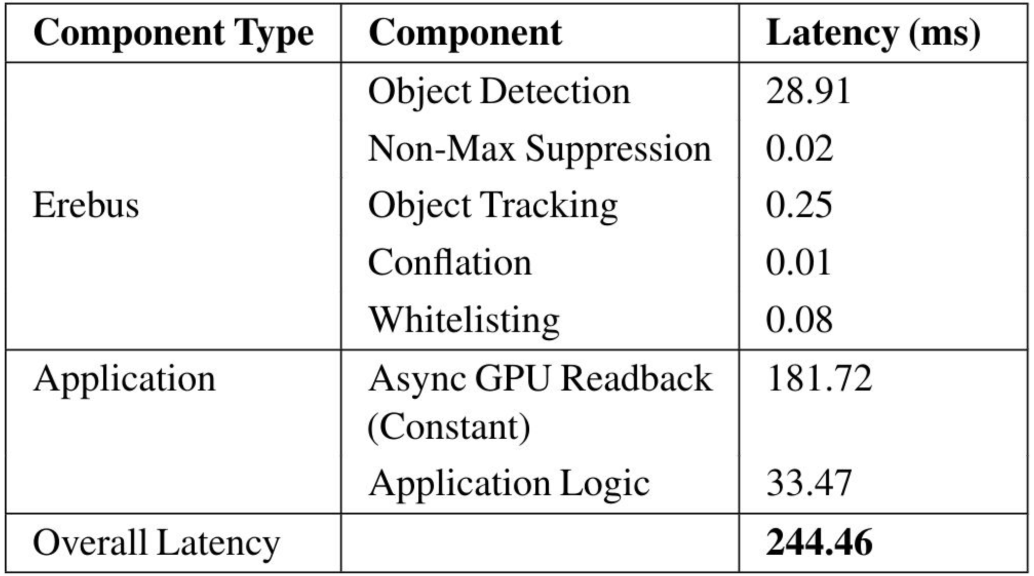
Does Erebus incur additional latency over API calls?



API Type	Erebus (ms)	Unprotected (ms)
Camera sensor-based API	0.35 ± 0.12	0.18 ± 0.04
Location sensor-based API	0.22 ± 0.04	<0.01

By enforcing runtime checks on API calls, there is a small overhead incurred by Erebus but this has no impact on performance.

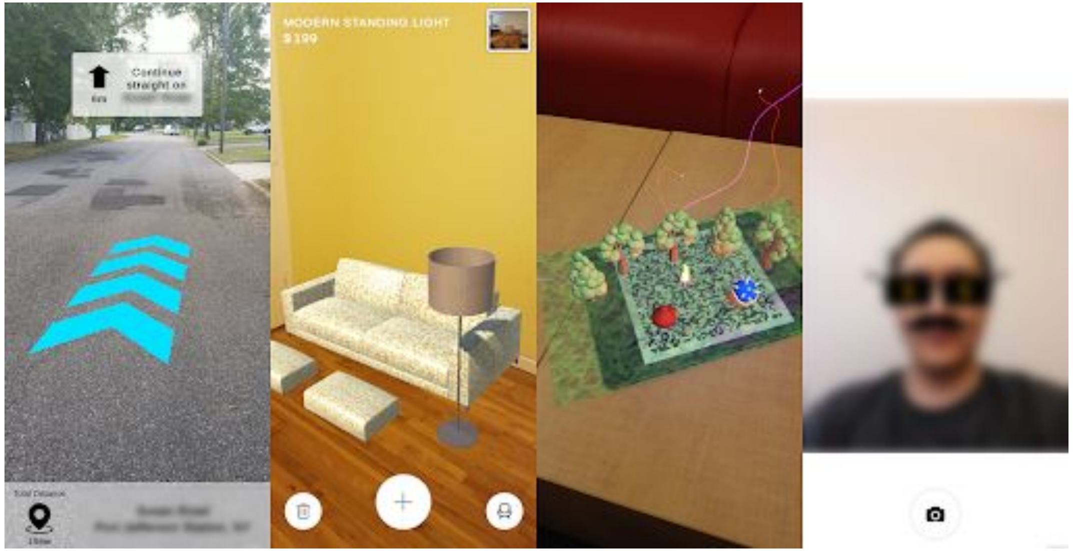
Does Erebus affect app’s overall performance?



Component Type	Component	Latency (ms)
Erebus	Object Detection	28.91
	Non-Max Suppression	0.02
	Object Tracking	0.25
	Conflation	0.01
	Whitelisting	0.08
Application	Async GPU Readback (Constant)	181.72
	Application Logic	33.47
Overall Latency		**244.46**

Our prototype apps were able to run at **~34.16 FPS** with Erebus framework enforcing runtime checks.

Erebus: adapting the framework



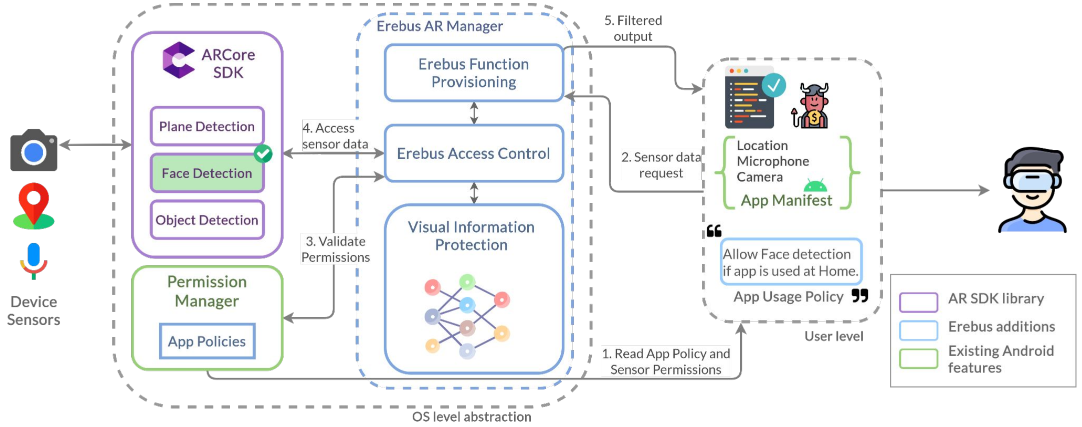
- Implemented on Google ARCore SDK using Unity Framework.
- Adapted 5 prototype AR applications to our framework.
- We open-source our framework implementation, policy-language design, and prototype applications for developer's reference.

ARTIFACT EVALUATED usenix ASSOCIATION AVAILABLE

ARTIFACT EVALUATED usenix ASSOCIATION FUNCTIONAL

ARTIFACT EVALUATED usenix ASSOCIATION REPRODUCED

Erebus: Access Control for Augmented Reality Systems



Sanket Goutam*, Yoonsang Kim*, Amir Rahmati, Arie Kaufman

Stony Brook University

https://github.com/Ethos-lab/erebus-AR_access_control https://sgoutam.github.io @sanketgoutam

* *equal contribution*