AR STICKERS

From Design to Launch

Ivan Neulander (ineula@google.com)
Daydream Team
Google, Los Angeles

What is AR Stickers?

- AR Stickers is a *camera mode* for Pixel & Pixel 2 phones.

- Place animated 3D stickers into the camera feed and they react to one another and to you.

- Stickers are lit and rendered to blend seamlessly into the scene.

Agenda

1. **Development Process**
2. AR Stickers Design
3. Lighting & Rendering
4. Visual Enhancements
5. Concluding Thoughts

Our Team

Many groups at Google collaborated to launch AR Stickers:

- Product Management (PM)
- Business Development
- Publishing Producers
- User Experience Designer
- User Experience Researcher } (UX)
- Visual Artists (VA)
- Software Engineers (Eng)
- Test Engineer
- Q/A Tester } (Q/A)

Our Team

Several external studios authored the final content:

- Meshes
- Textures
- Skeletal Animations
- Sounds

Team Photos

Mountain View

Google Daydream

Los Angeles

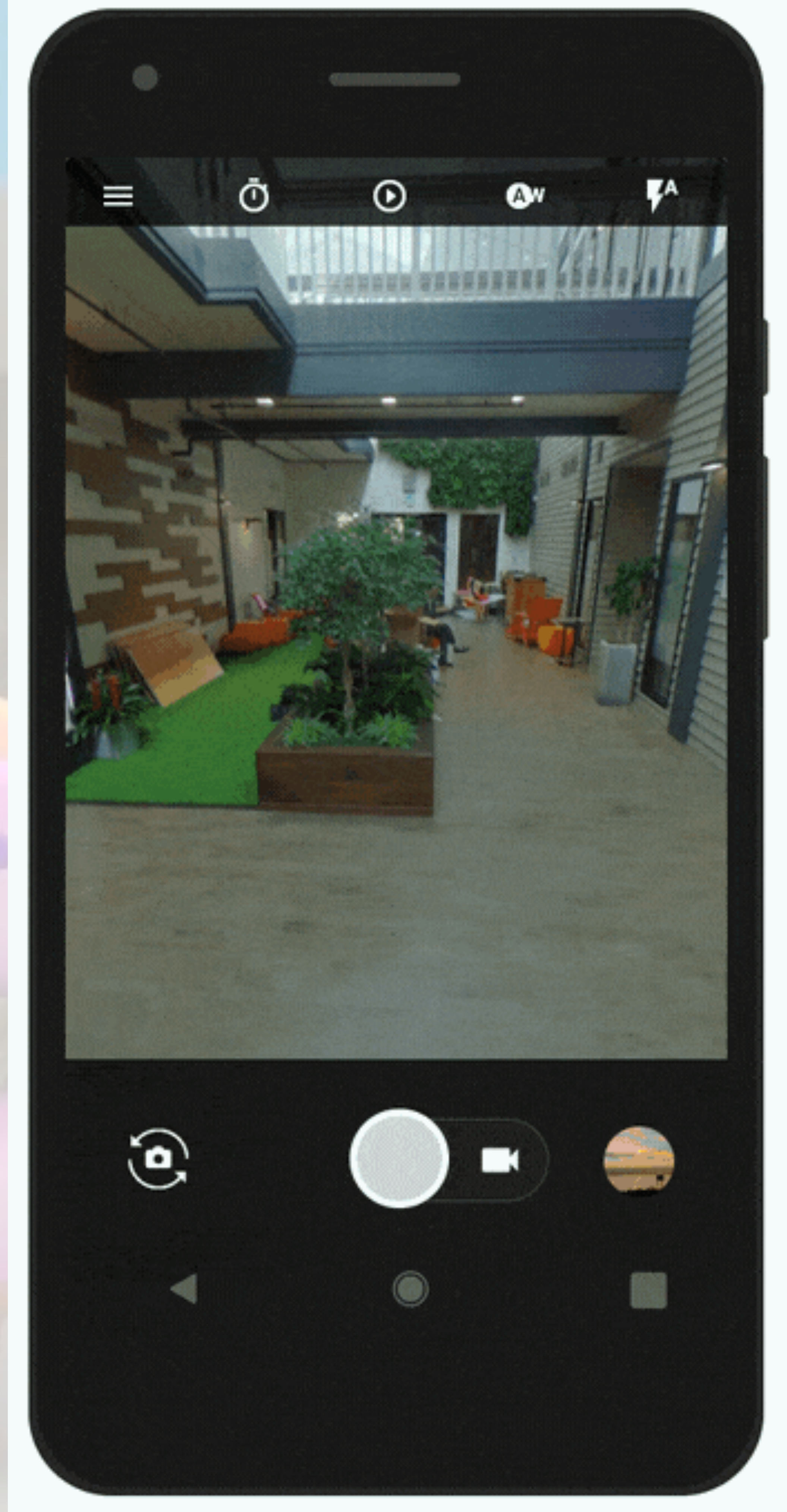
Team Photos

Google
Daydream

Design Mockups

UX provided mockups to guide initial design.

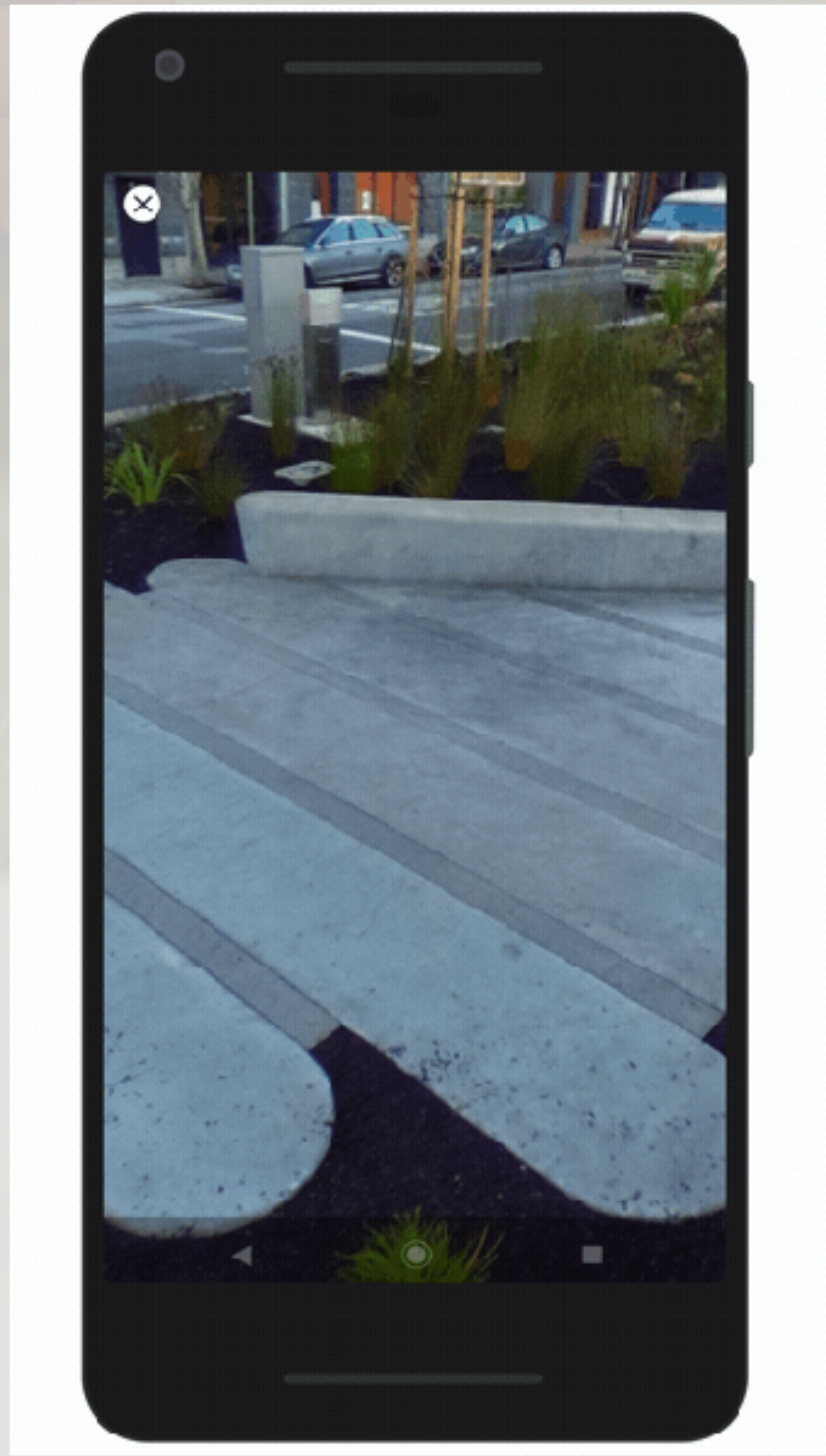
- Defined basic layout and behaviors.
- Completed in advance of implementation by engineers.

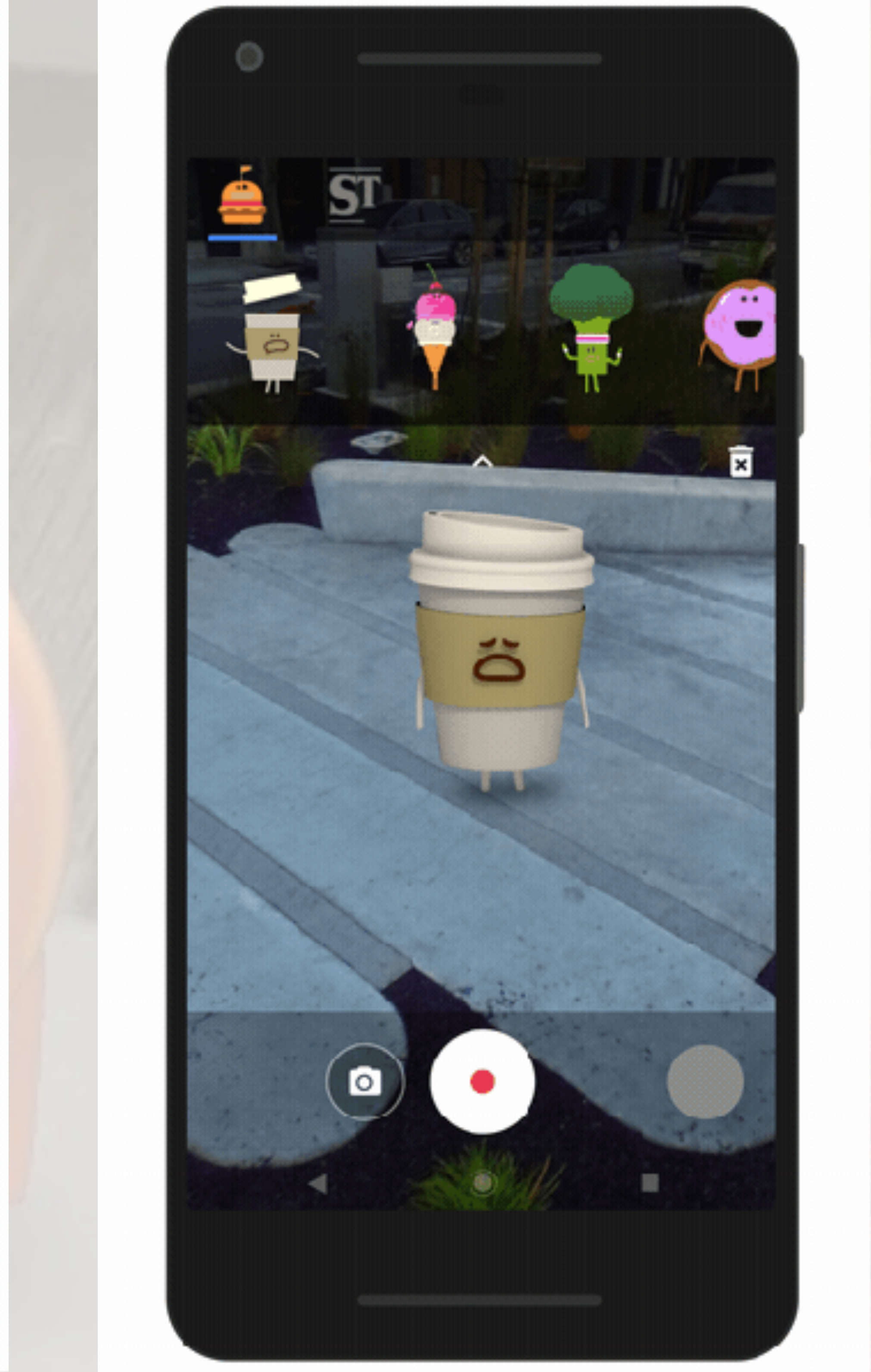


Design Mockups

UX refined the interface as new features were added.

- Iterated with Eng to arrive at the final look.
- Some ideas came later, e.g:
  - Tutorial animation.
  - Ground dots.
  - Out-of-bounds reticle.





Prototype

Why build a prototype?

- Validate design & justify further development early on.
- ARCore and 2017 Pixel weren’t yet available:
  - Emulated on Tango phones with Unity plugin.

Tango is the precursor to ARCore (uses special depth sensors).

Prototype

Built in Unity

- Lots of built-in functionality => fast prototyping.
- Many platforms, including Android.
- C# scripting + native plugins.
- Flexible animation system.
- Physically-Based Rendering shaders.

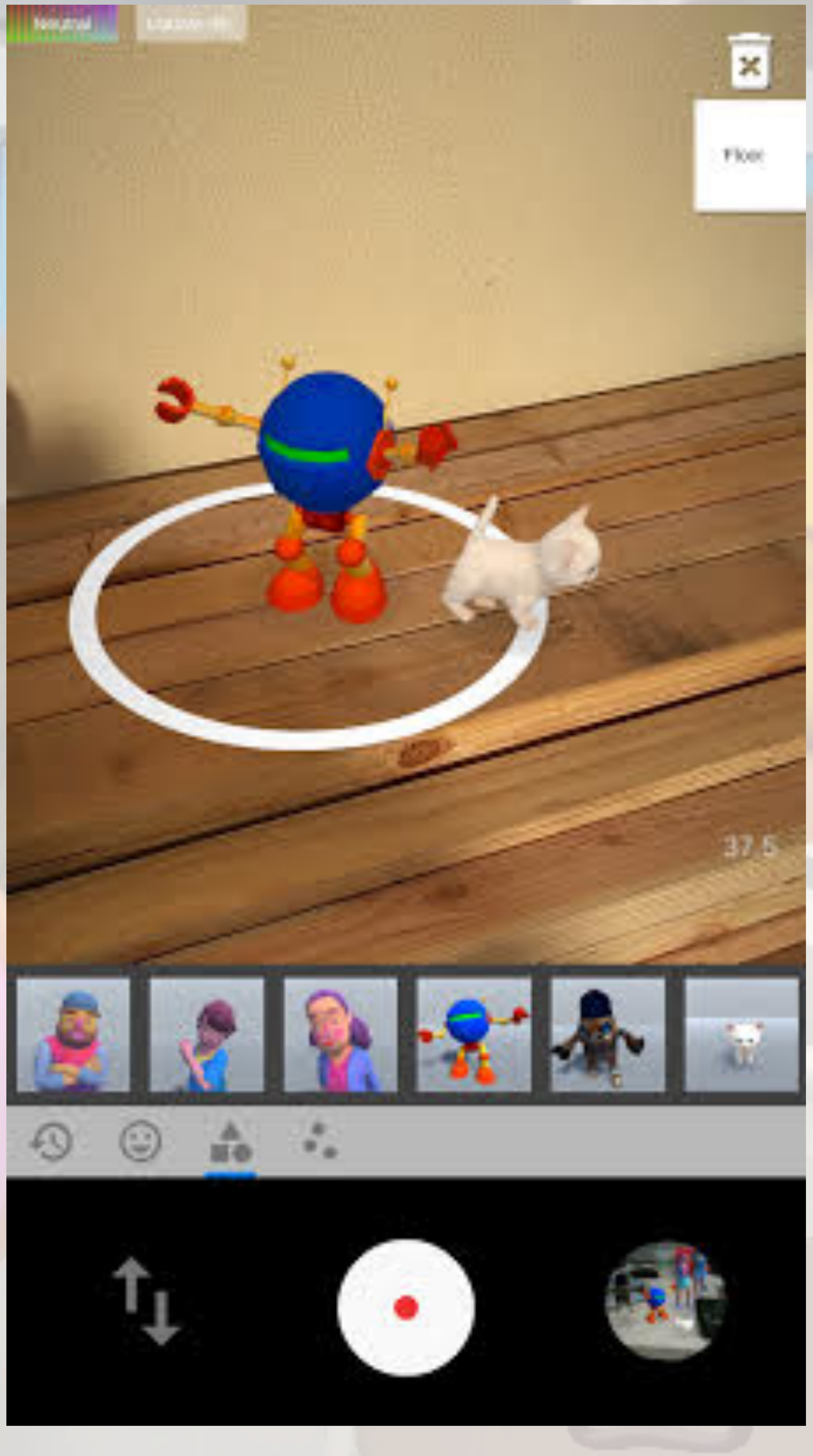
Prototype

Got the basics working.

- Placement, deletion, translation.
- Icons, reticles, gestures.
- Lighting.
- Video recording.

Used Tango to emulate ARCore.

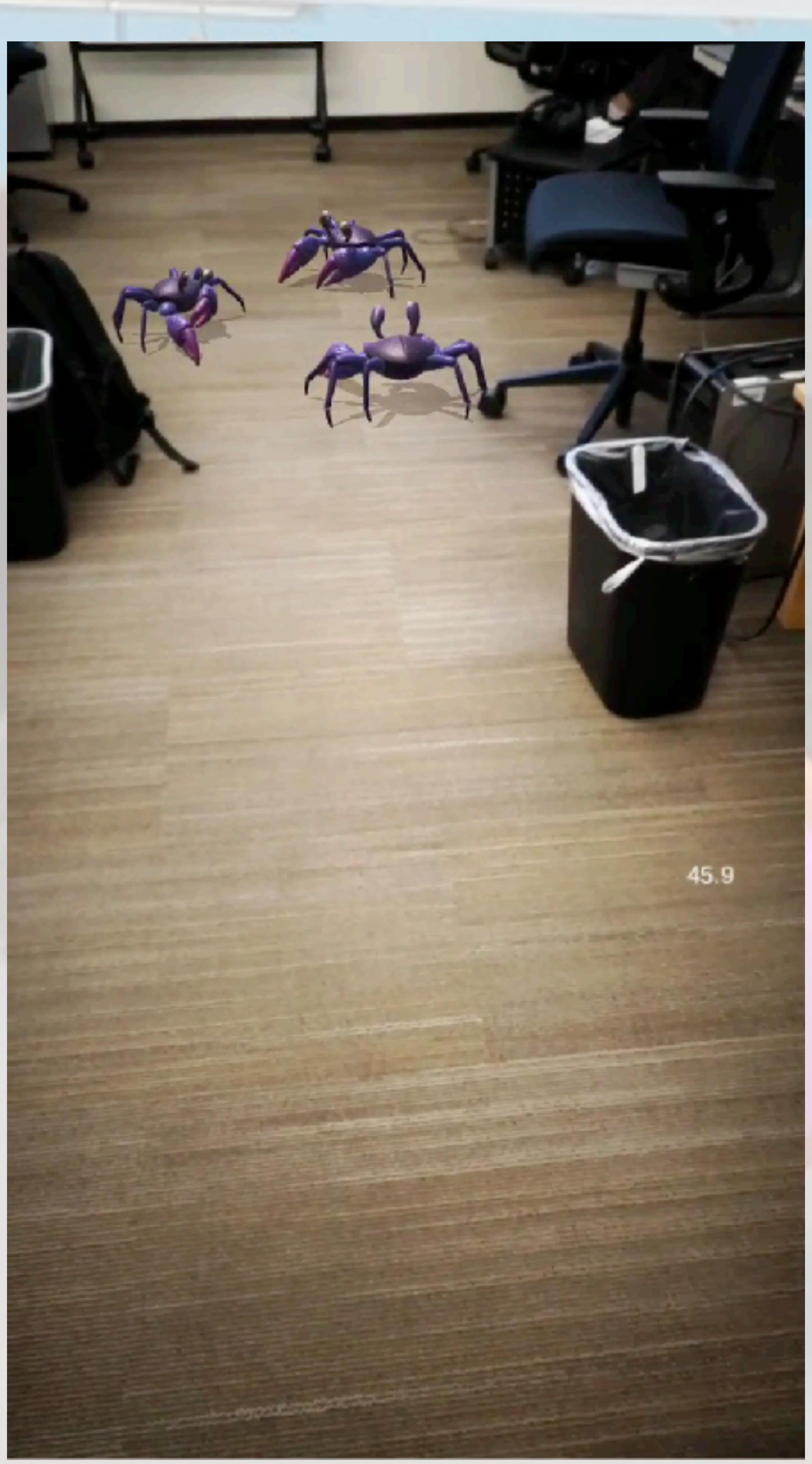
- Ran on existing Tango phones.
- Manual camera tracking.
  - User follows feature point with phone.

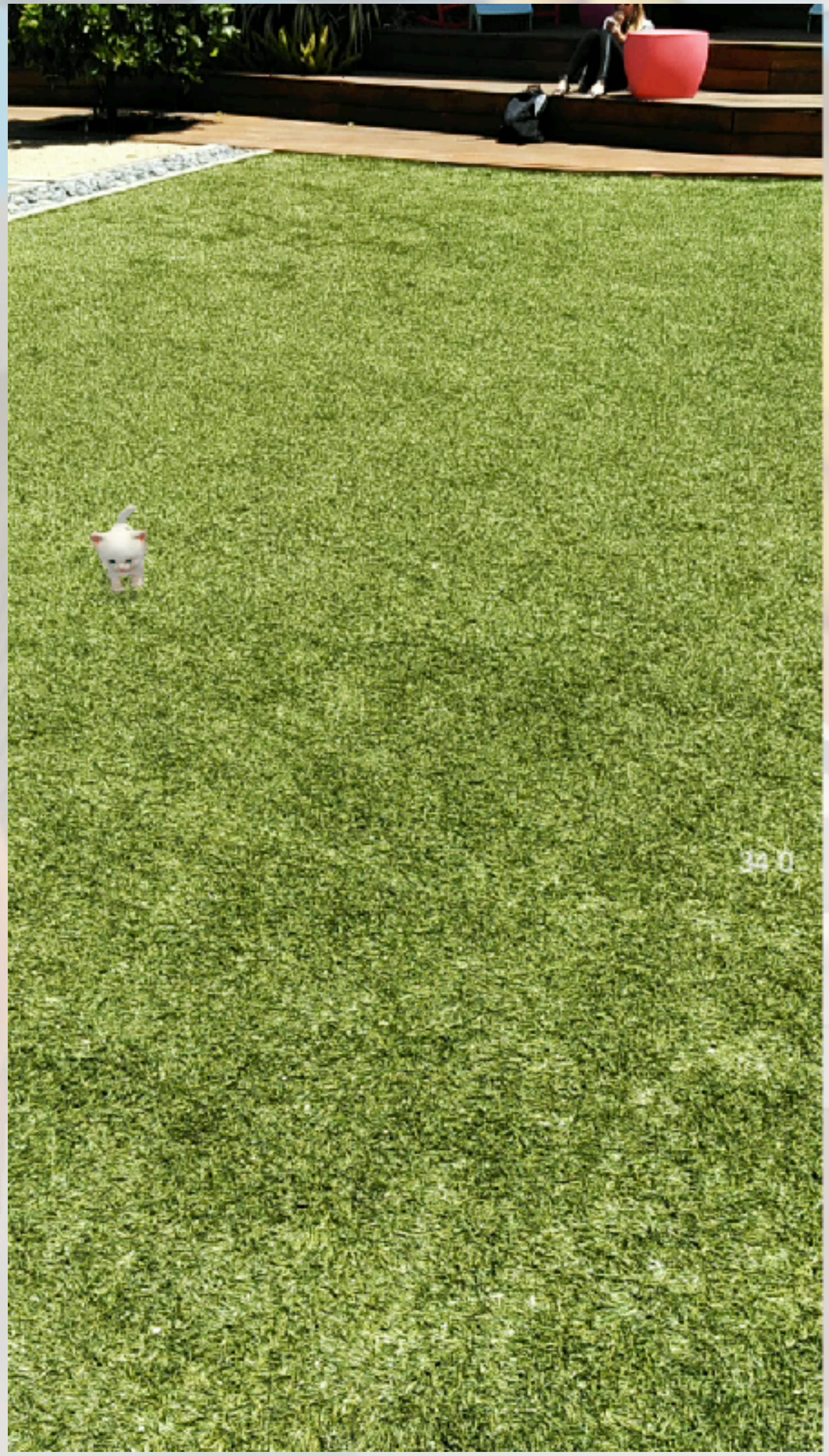


Prototype

Basic interactive animations.

- User-to-sticker proximity.
- Sticker-to-sticker proximity.



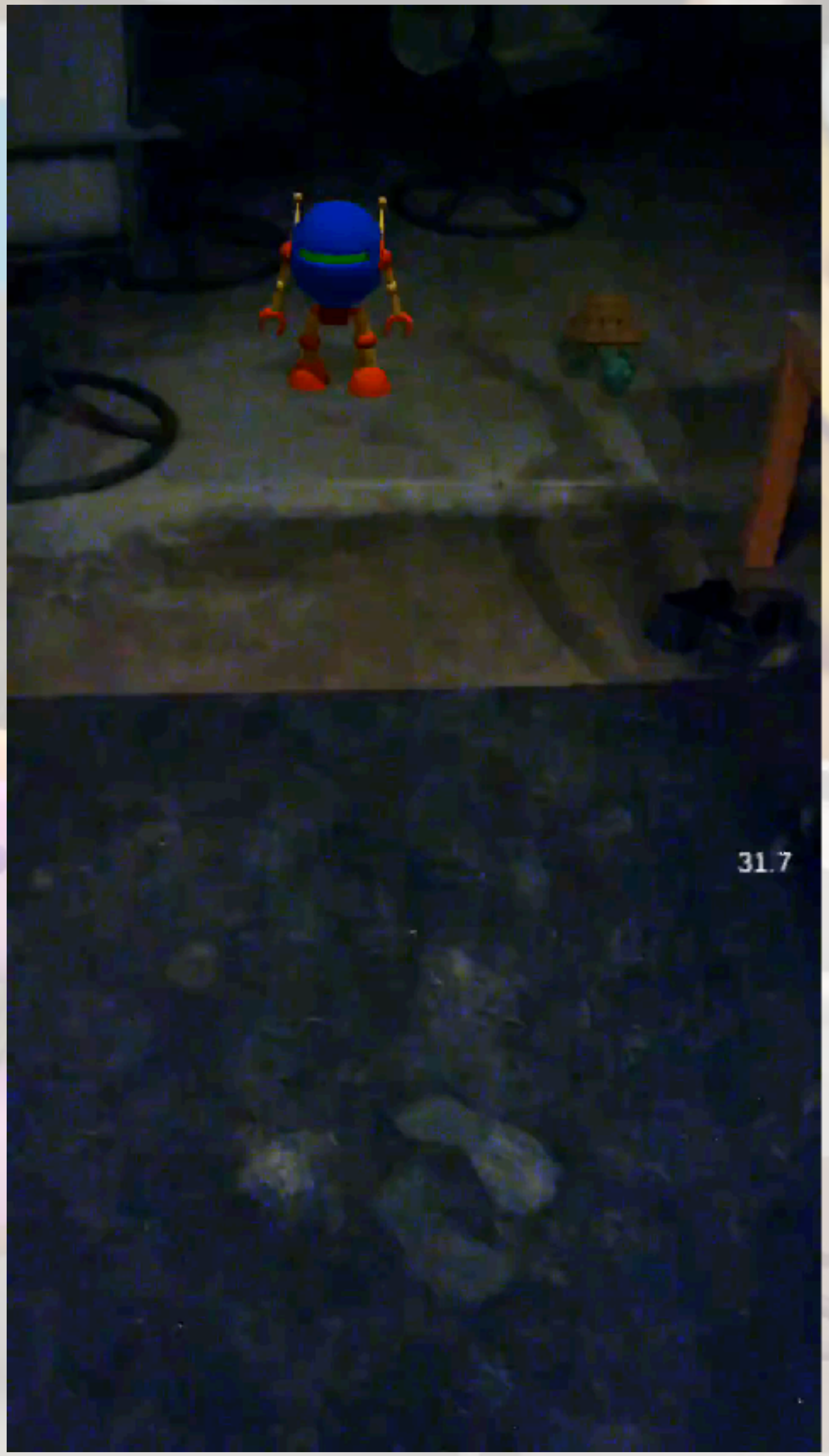


Graveyard Battle

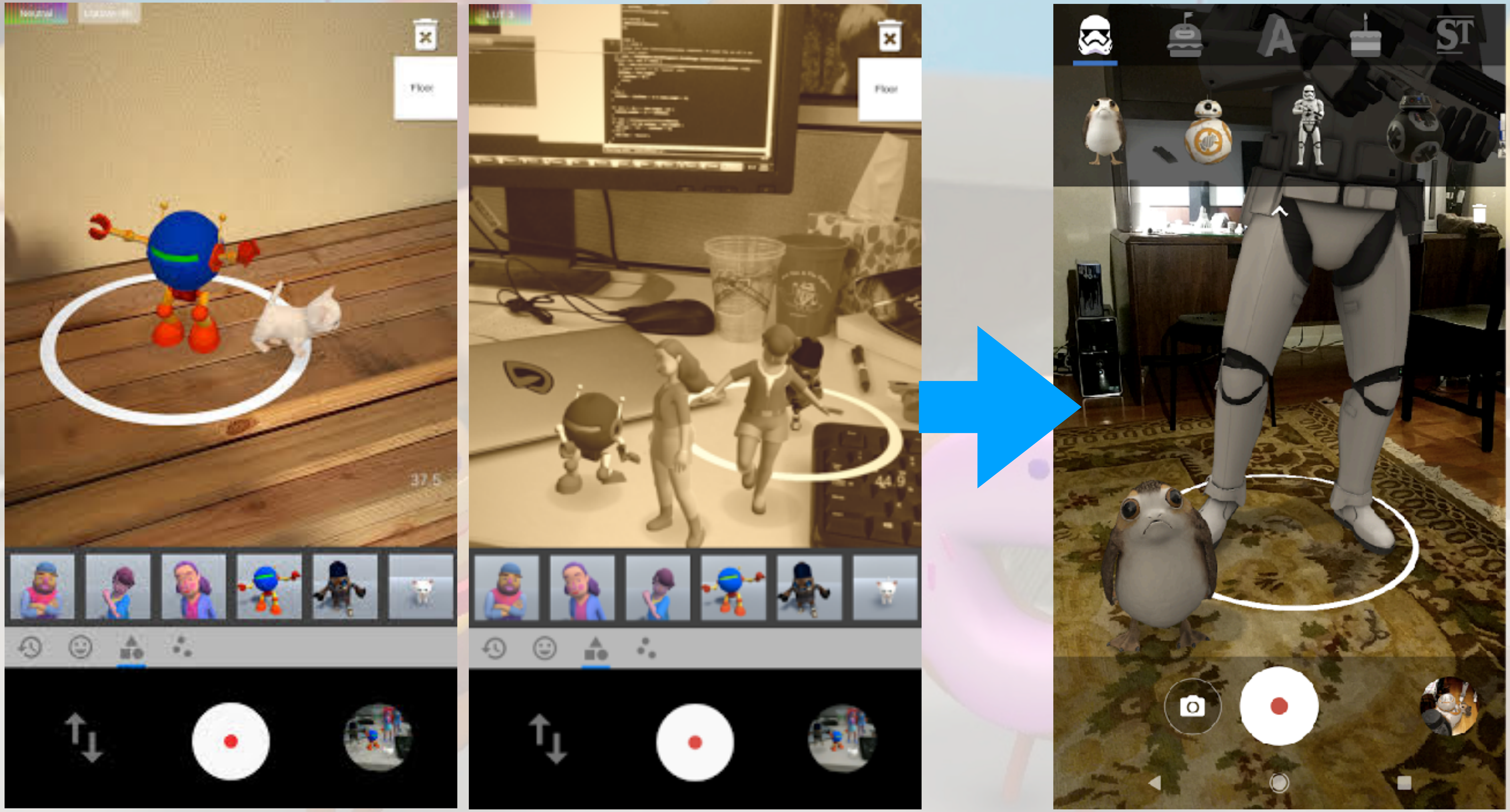
Prototype

Illumination effect.

- Stickers can light the real world.
- Uses ground plane color scaling + halo effect.
- Would have benefited from more tracked geometry (e.g. horizontal + vertical planes).



Final Product



Final Product

Total rewrite of the Unity prototype.

- Event loop, UI, screen recording in Android / Java.
- Animation, rendering, sound in ***Lullaby.***
- AR tracking in ***ARCore***.

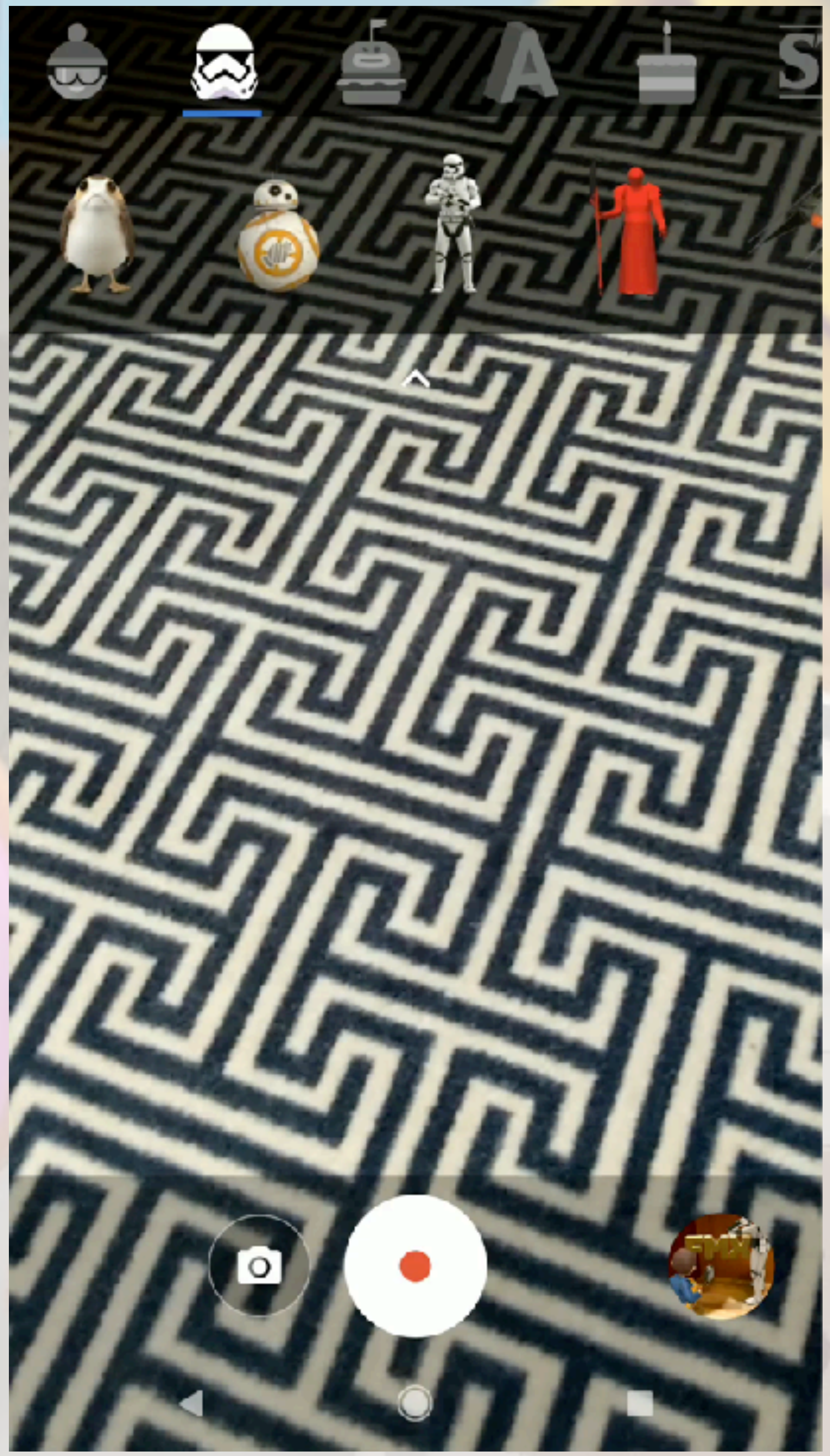
Tight Schedule

- From prototype to public demo in < 4 months.
- Launched in December 2017

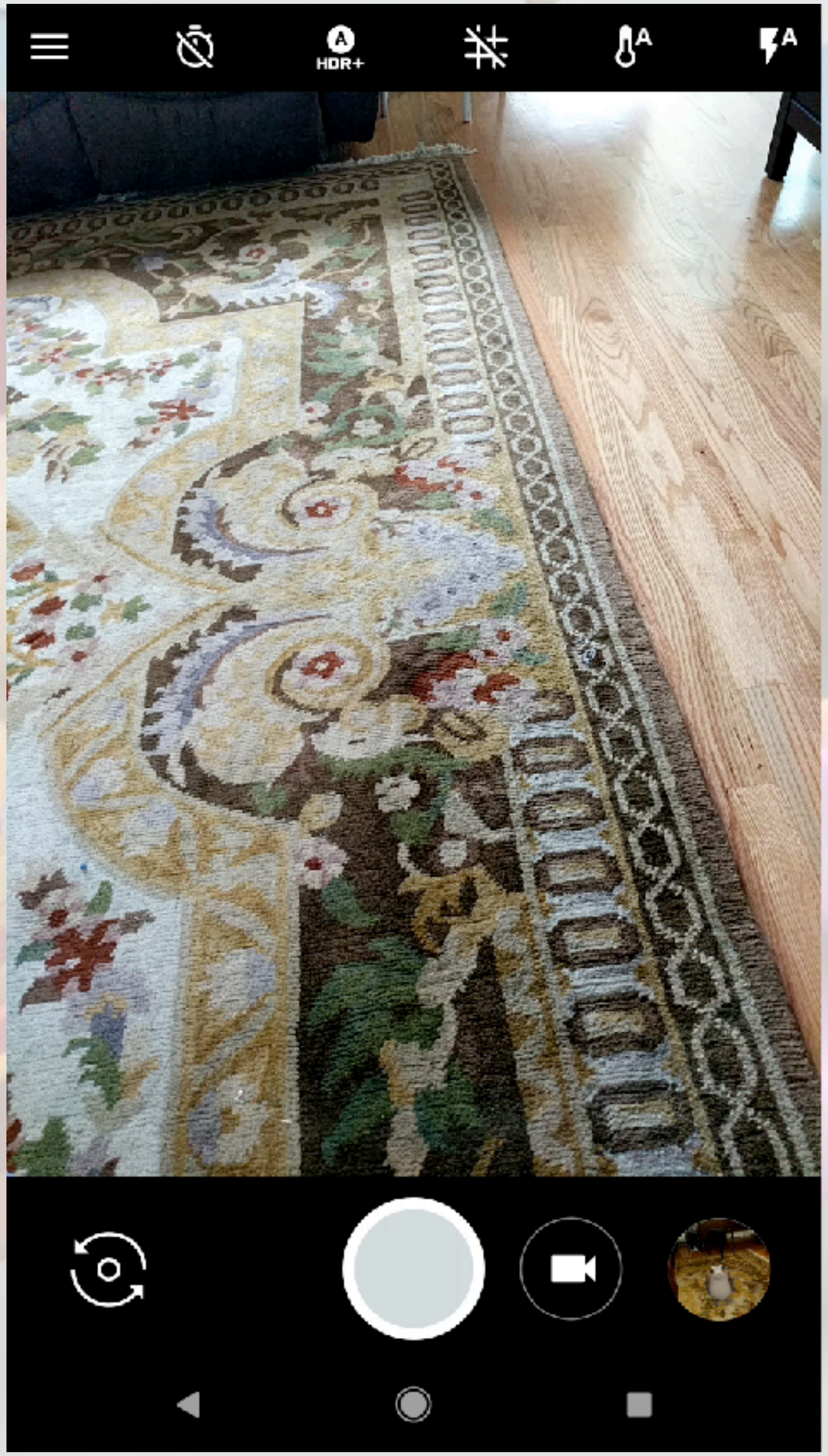
Final Product

Sticker assets formatted by our build pipeline.

- E.g. **ASTC/KTX** textures.
  - Vastly reduced load time vs. WebP compression.



Quick Demo



Agenda

1. Development Process
2. **AR Stickers Design**
3. Lighting & Rendering
4. Visual Enhancements
5. Concluding Thoughts

Lullaby

AR Stickers needed:

- Smaller APK.
- Faster startup.
- More customizability.

Lullaby is an open-source multi-platform engine for VR + AR.

- https://github.com/google/lullaby

Lullaby

Why we chose Lullaby:

- Specifically designed for mixed reality.
- Originated at Google:
  - Easy access to latest source code + dev team.
  - Good integration with our standard build system and dev tools.
- Extensible codebase (lightweight C++ libraries).

Google Daydream

ARCore

https://developers.google.com/ar/

ARCore is Google's Open-Source Augmented Reality SDK for: **Android, Unity, Unreal, the Web.**

- **Provides per-frame estimation of**
  - Camera pose (position + rotation).
  - Visible planar surfaces (includes boundaries).
  - Scene lighting.
- **Allows object to be anchored to feature point**
  - Its tracking improves over time.
- **Includes a C and a Java API.**

ARCore

No specialized hardware, just camera + IMU.

- Flagship phones by Google, Samsung, LG supported.

Runs continuously during video capture in AR clients.

- Analyzes downsampled video feed.
- Delivers results quickly, refines them over time.
- Modest CPU / battery usage.

Camera calibration helps with accuracy.

- Pixel 2: individual calibration.
- Pixel: batch calibration.

ARCore: Newer APIs

ARCore added newer APIs that we included after AR Stickers 1.0 launch:

Feature Point Clouds

- AR Stickers uses these for a quick initial estimate of ground plane, based on median of y-values, limited to some range.
- Work quite well with a single ground plane.

ARCore: Newer APIs

ARCore added newer APIs that we included after AR Stickers 1.0 launch:

Resumable Sessions

- Restores existing stickers + tracking after leaving app.
- Important when recording, sharing, returning to app.
- Assumes that phone doesn’t move much while ARCore is dormant.

Google
Daydream

User Interface

Mobile AR is fraught with optical illusions.

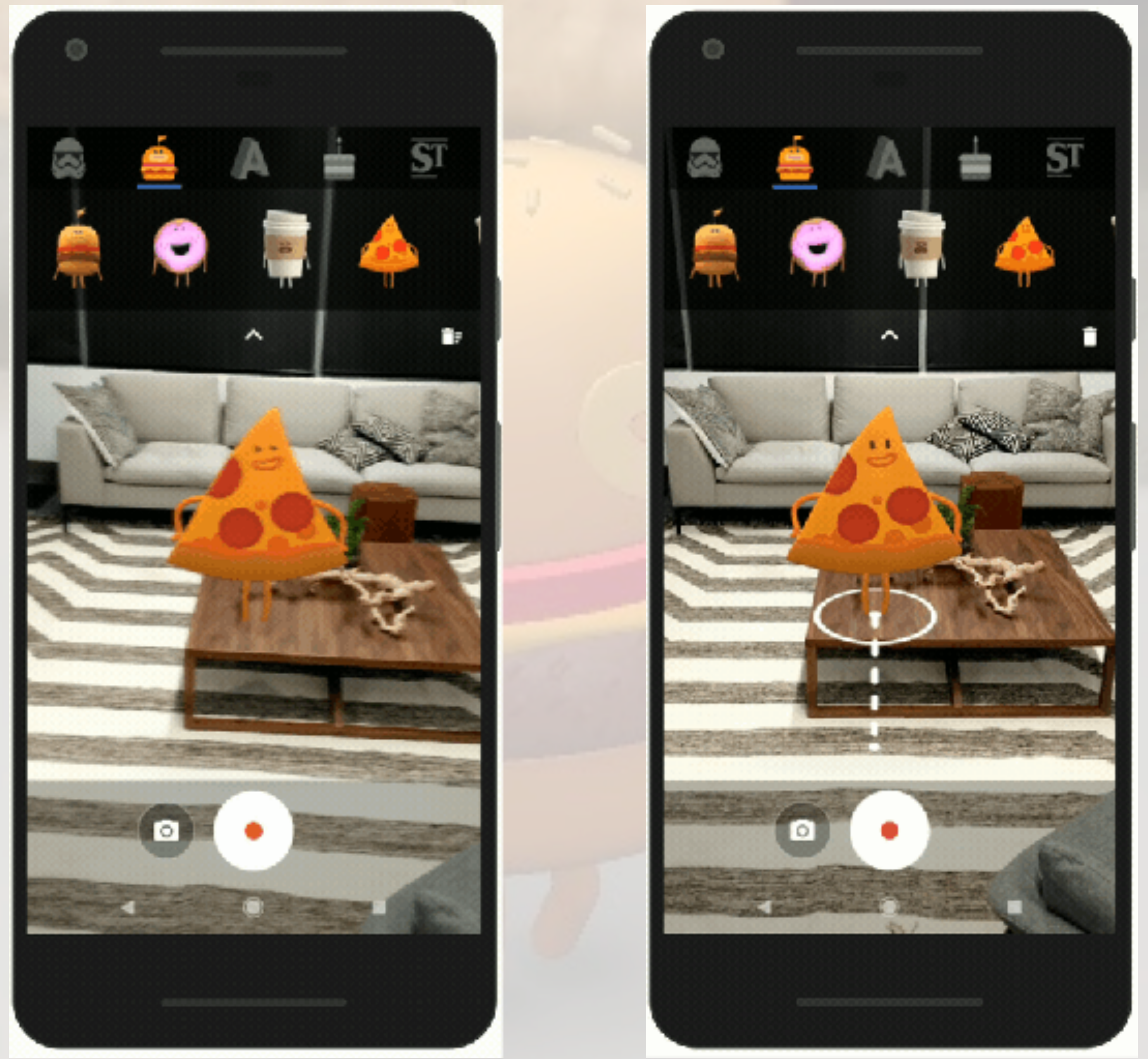
- 3D objects in a 2D view can be ambiguous.

- Our goal was to:
  - break optical illusions
  - ground characters
  - create visual references so users can intuit where their objects are placed in the world.

User Interface

One Example:

- With the UI reticle, we immediately see that Pizza is not on the table, but is floating above it.

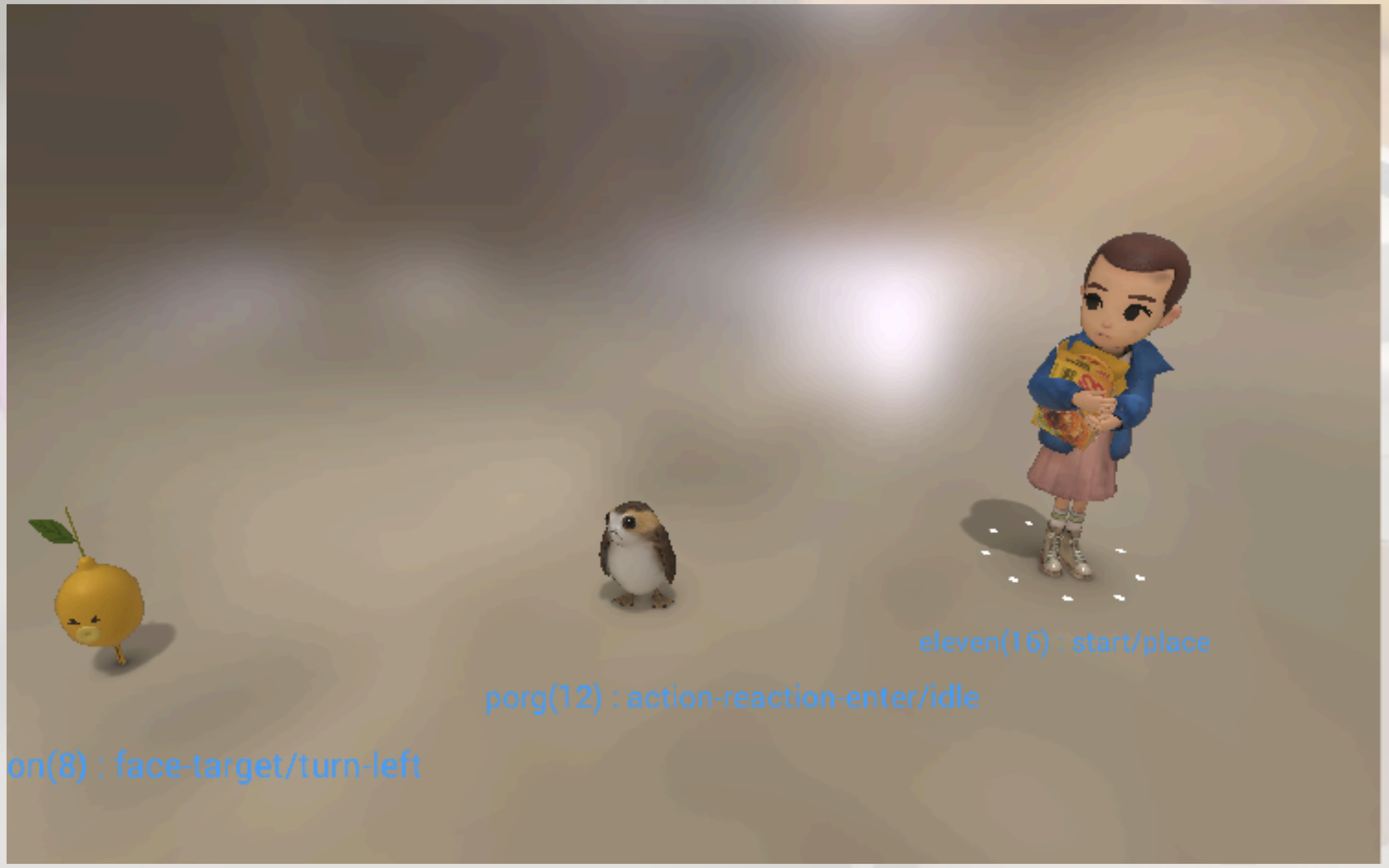


Agenda

1. Development Process
2. AR Stickers Design
3. **Lighting & Rendering**
4. Visual Enhancements
5. Concluding Thoughts

Desktop Viewer

- Same engine as the app.
  - Runs on Linux, Mac, Windows.
  - Consistent rendering, animations, sounds.
- Faster test iterations than on Android.
- VAs & studios can validate content.
  - Useful for engineers too.



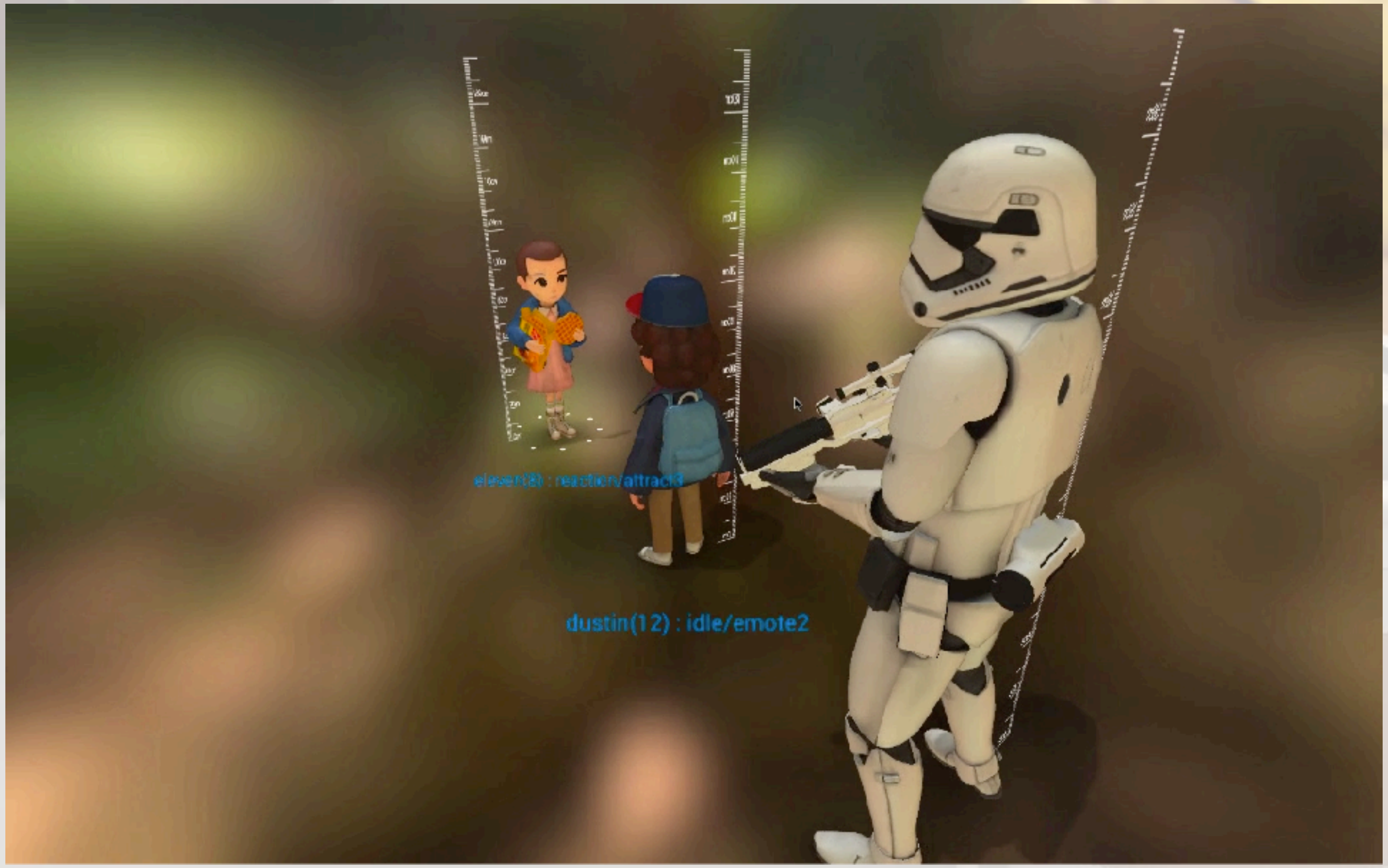
eleven(16) : start/place

porg(12) : action-reaction-enter/idle

on(8) : face-target/turn-left

Desktop Viewer: Features

- Switching environments.
- Pausing animations.
- Ruler.
- Altering the shader's
  - Albedo.
  - Smoothess.
  - Metalness.
  - Emissiveness.



Desktop Viewer: Simulated Camera Feed

ARCore tracking embedded in a JPEG.

- Metadata stored as EXIF.
- Simulates on-device capture.
- Makes viewer behave more like our app (some UI differences).

- Currently only in viewer, but could be useful as in-app experience.

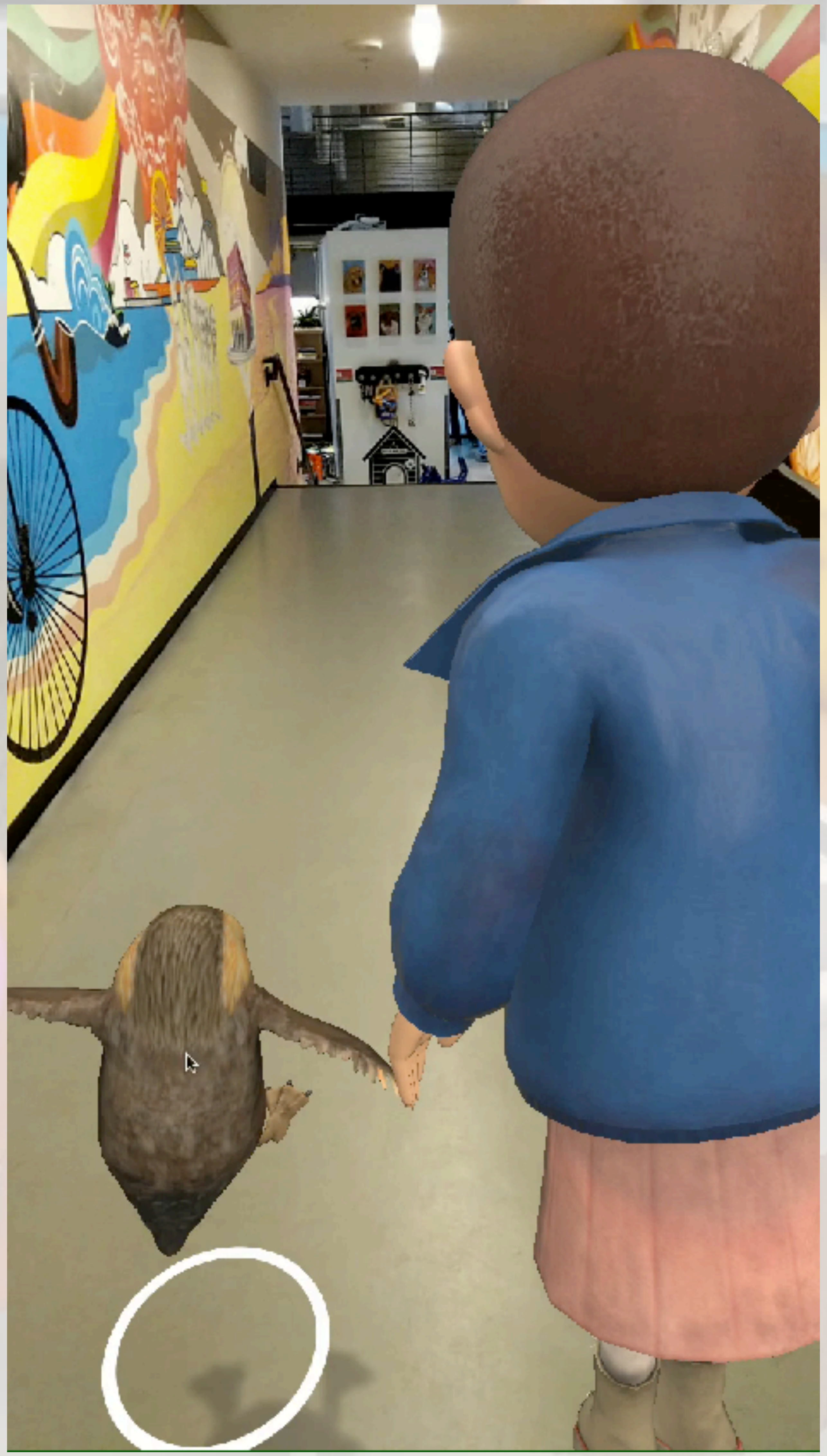


Image-Based Lighting

- Prototype used traditional directional lights.
  - Objects appeared harshly lit.
  - Abrupt transition between light and shadow.

- Motivated transition to IBL.

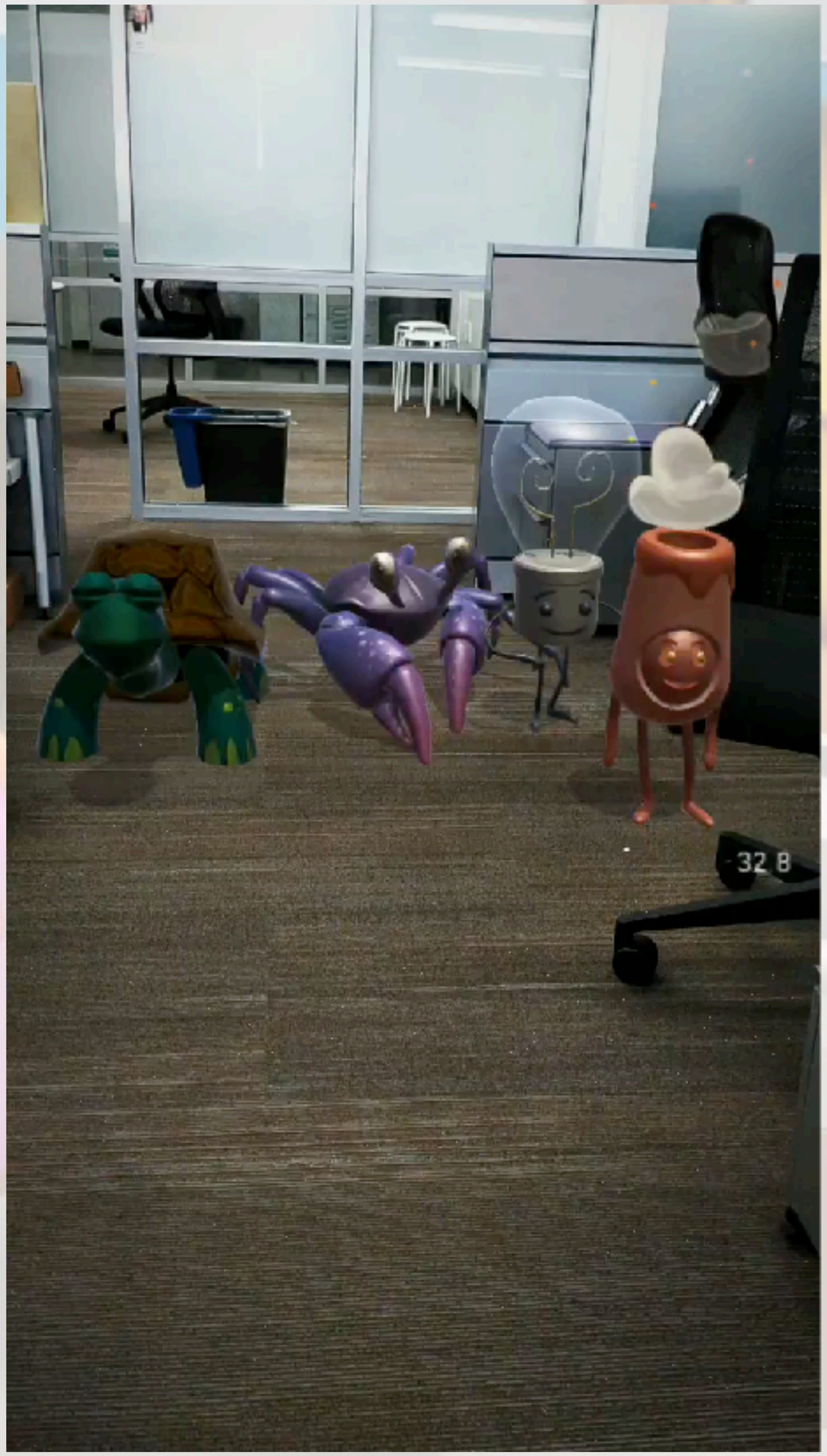
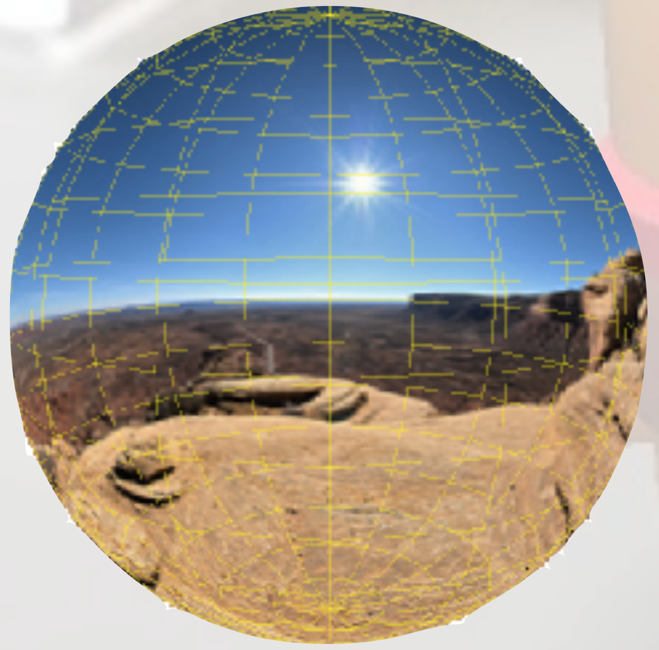


Image-Based Lighting

Q: What do we mean by **Image-Based Lighting**?

A: We use a cube map to illuminate our CG elements.

- Compactly represents a complex lighting environment.
- Techniques for efficient rendering.
- We can alter lighting by changing cube map.



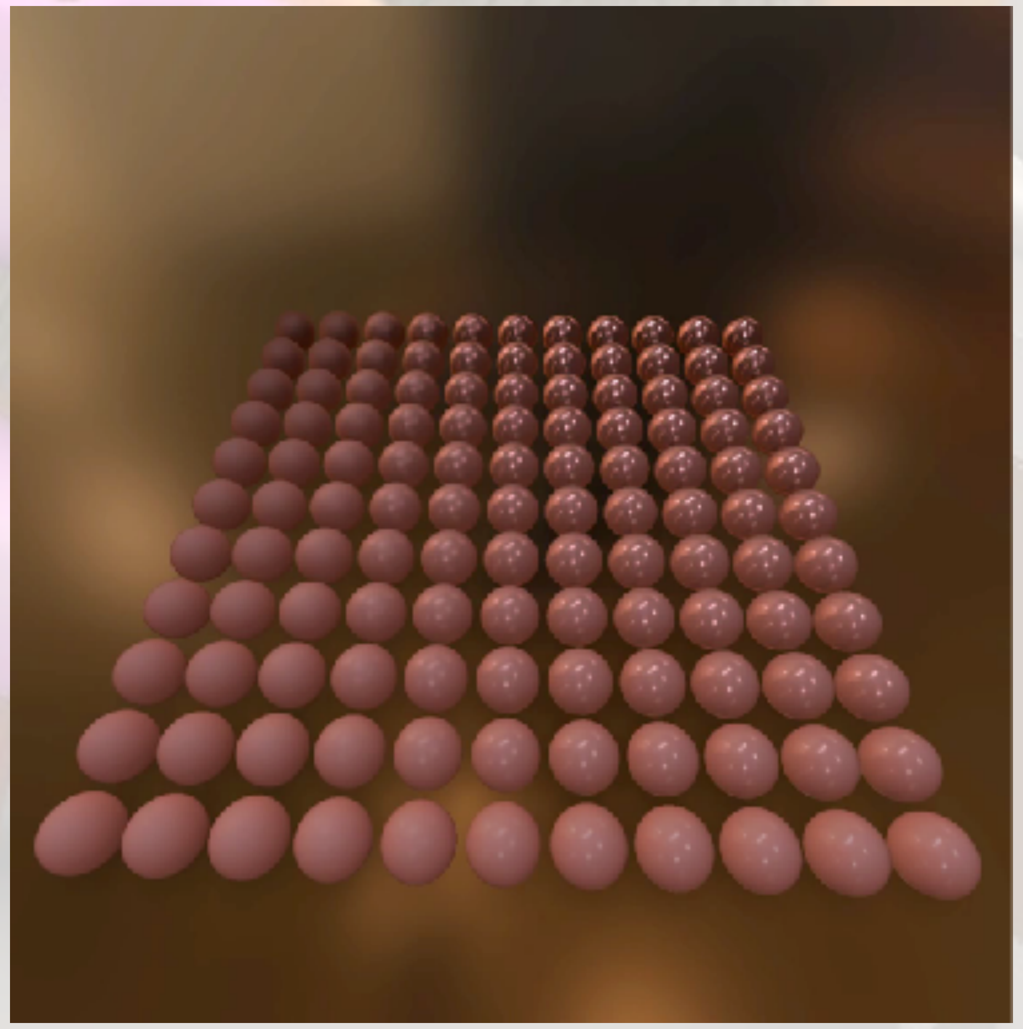
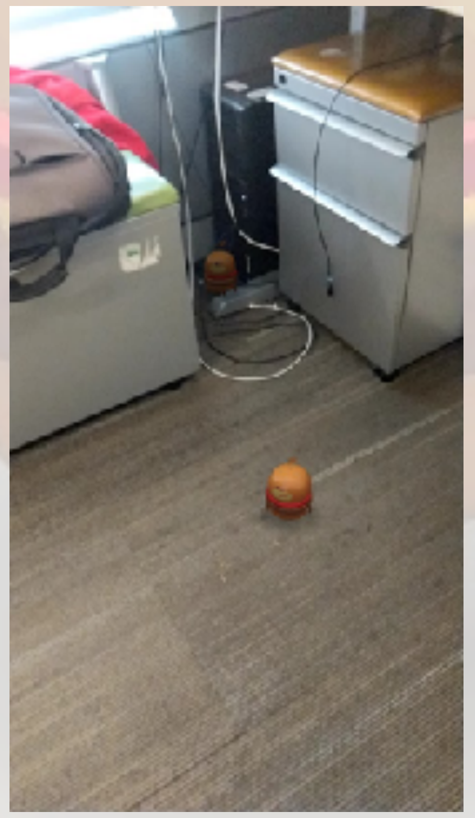
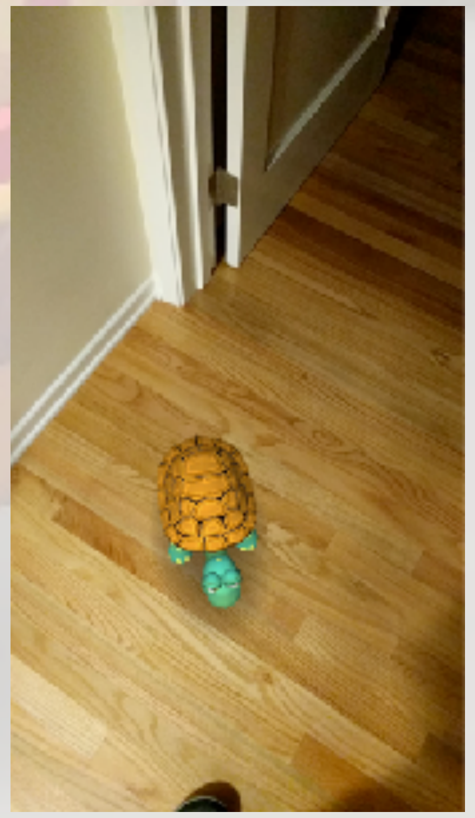


Image-Based Lighting

How do we use IBL in AR Stickers? Two basic components:

1. Precomputed diffuse + specular reflection from an existing HDRI panoramic image.

2. “Fake” Image-Based Lighting uses camera feed.

IBL: Precomputed Lighting

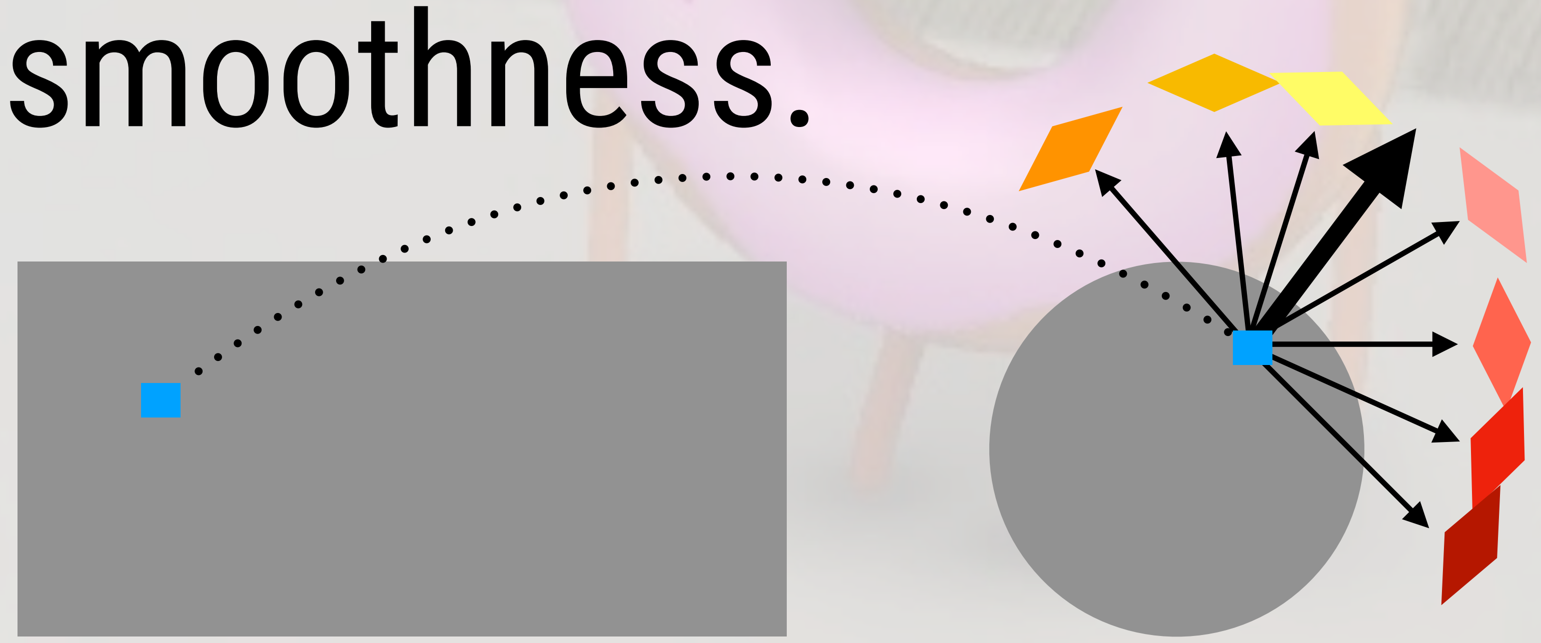
Miller & Hoffman [1984]

- Lighting baked to texture.
- Very inexpensive at runtime.
- But: Ignores occlusion.

IBL: Precomputed Lighting

Precomputing Reflections from HDRI Panorama

- Separate *preconvolved* textures for diffuse + specular.
- Both are based on a weighted average of all incident light. from env sphere onto given point on a lit sphere.
- Weighting term allows variable falloff: $\cos^n \theta$
- Diffuse: indexed by surface normal, *n* = 1.
- Specular: indexed by reflection vector.
  - *n* value comes from shader smoothness.
- We handle varying *n* using texture blur (mip LOD bias).

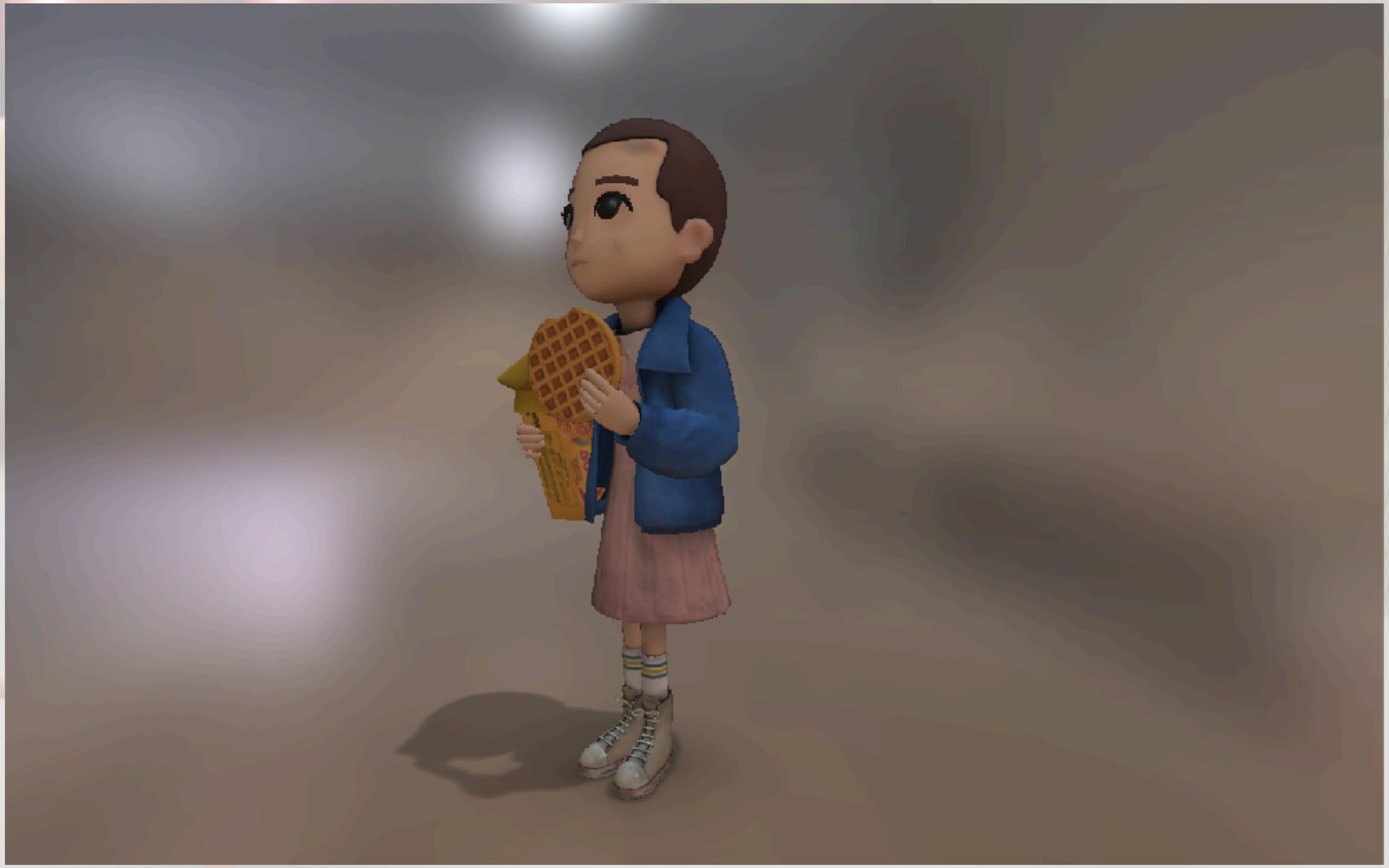


$$\sum_{\Omega} C_i (\vec{\omega}_i \,.\, \vec{N})^n$$

IBL: Precomputed Lighting

IBL: Precomputed Lighting

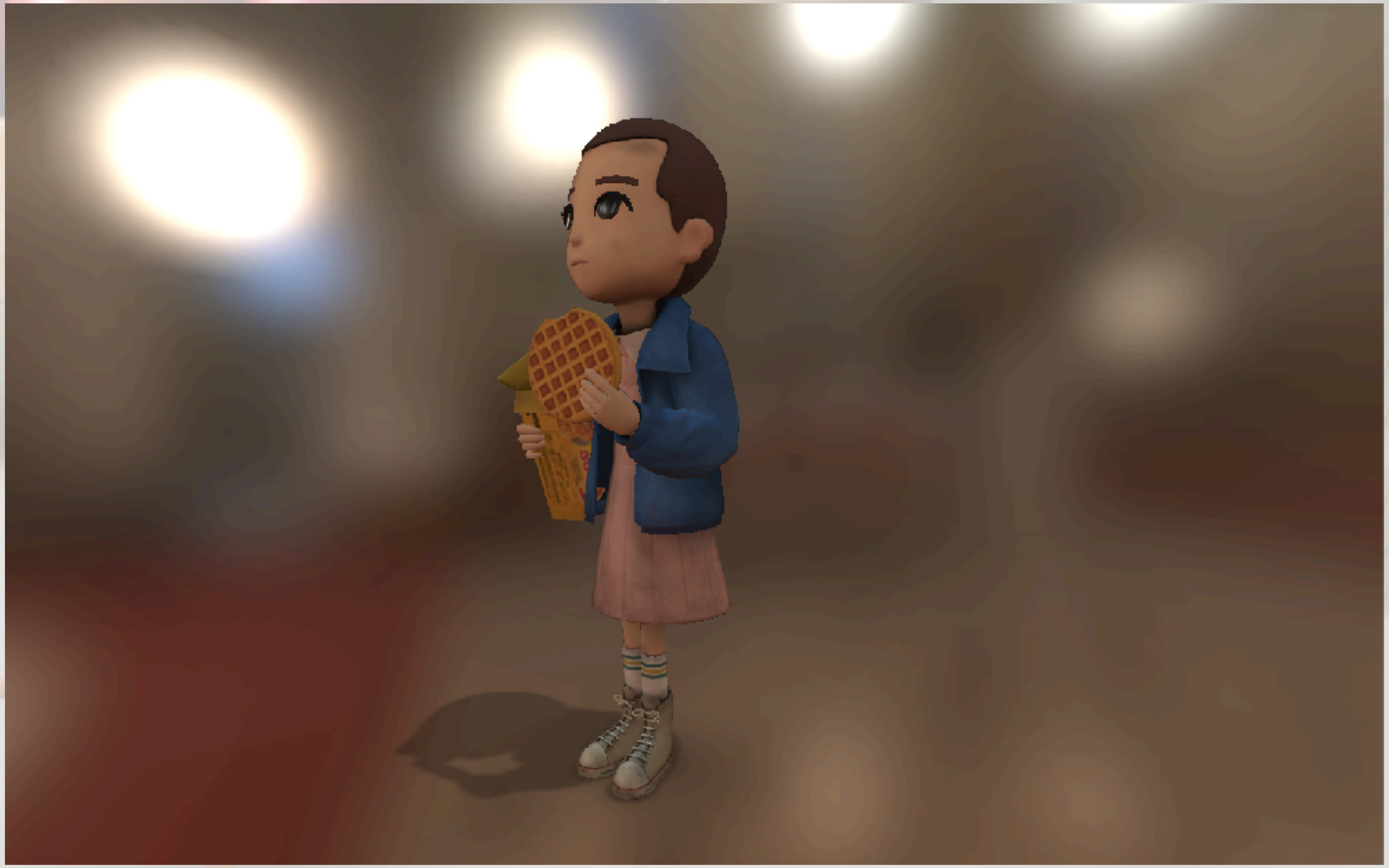
IBL: Precomputed Lighting



IBL: Precomputed Lighting

IBL: Precomputed Lighting

IBL: Precomputed Lighting



IBL: Precomputed Lighting

Precomputing Reflections from HDRI Panorama

- Looks pretty.
- Provides visual detail and interest.

But

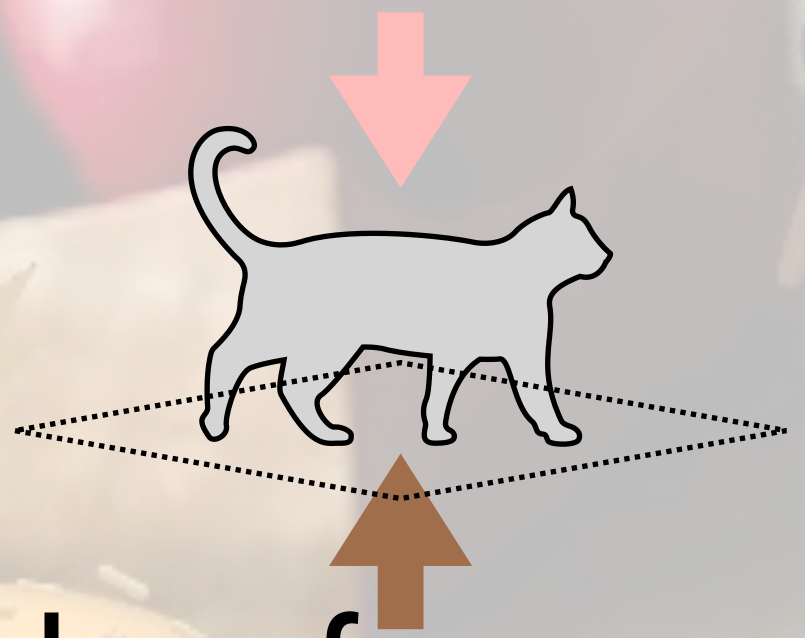
- Ignores phone camera, so lighting doesn’t match reality.
- Does not account for occlusion.

IBL: fake Image-Based Lighting

- Incorporates images from phone's camera feed into the lighting.
- Lacks detail, since phone's camera only sees a tiny part of environment.
- Is an inexpensive estimate of actual environment lighting (but usually plausible).
- Complements the precomputed HDRI reflections.

IBL: fake Image-Based Lighting

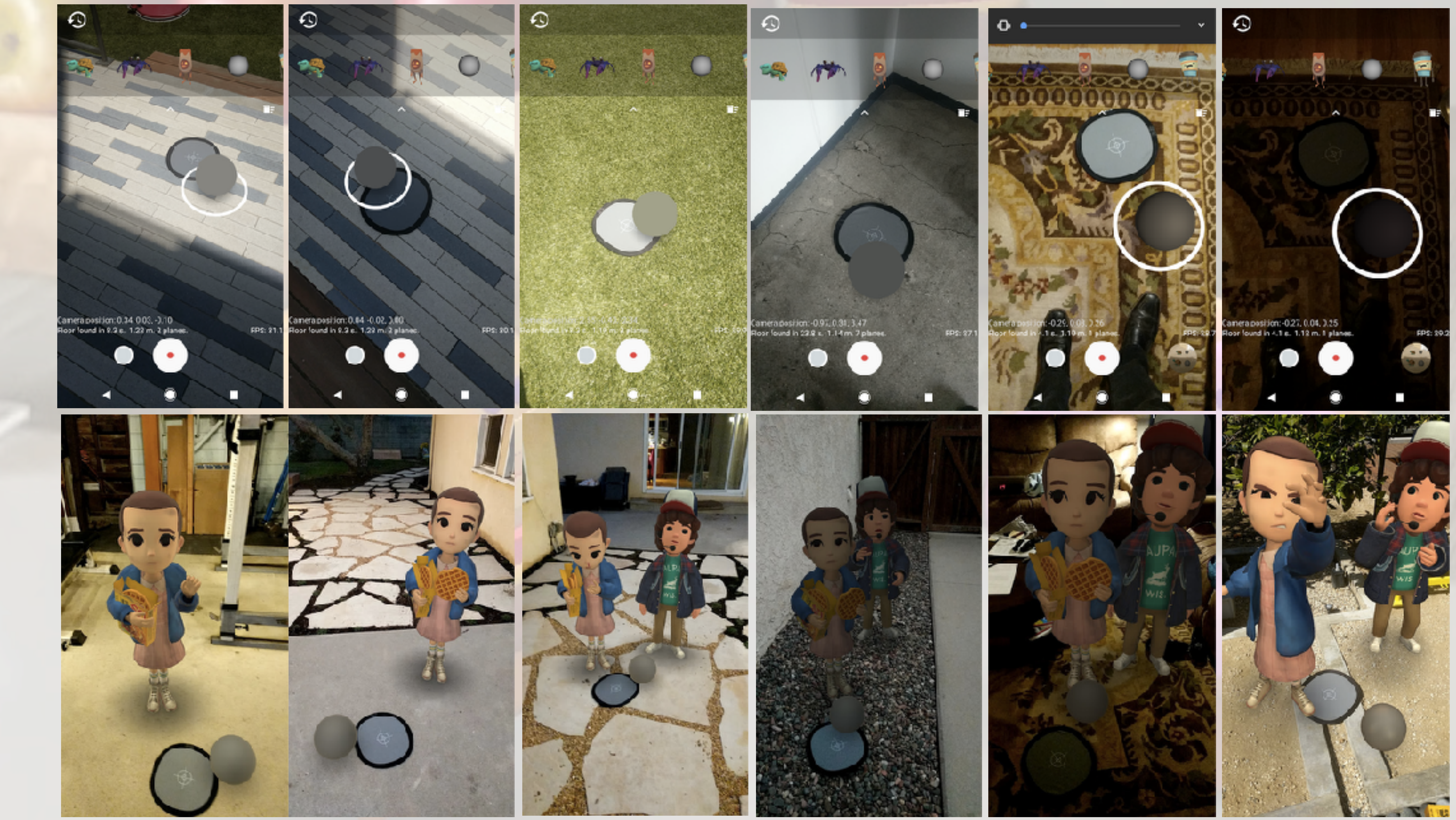
- Uses blurry, downsampled copy of camera feed.
- Samples region around bottom of sticker, separated into light from above and below:



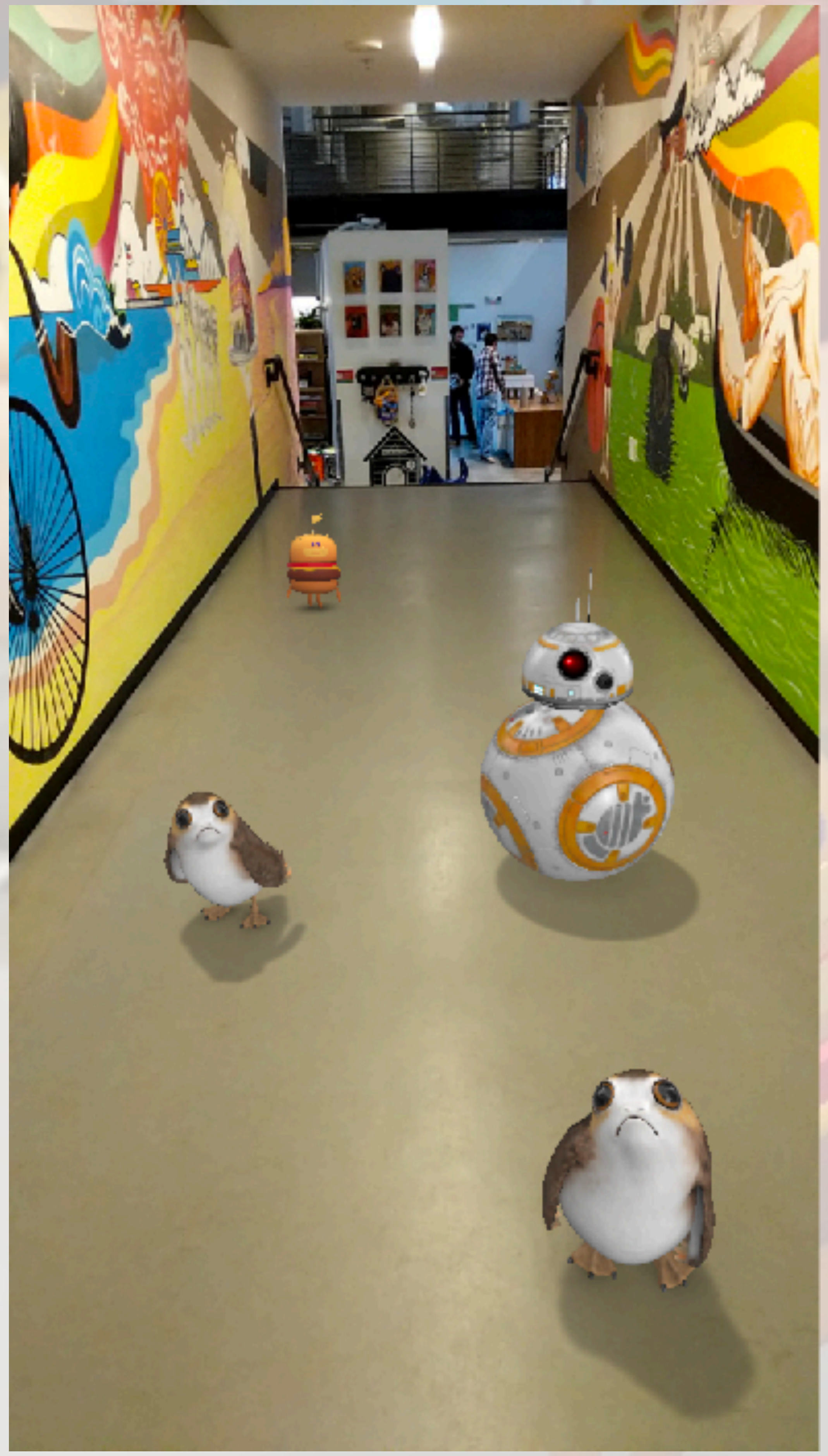
  - **Below**: Filtered color of camera view near sticker bottom (i.e. floor) is generally accurate.
  - **Above**: Also based on floor near sticker, but broader filter area and more desaturated. Affected by floor color.
- Drives procedural environment map with above/below colors.
  - Scaled with precomputed lighting lookups.

IBL: fake Image-Based Lighting

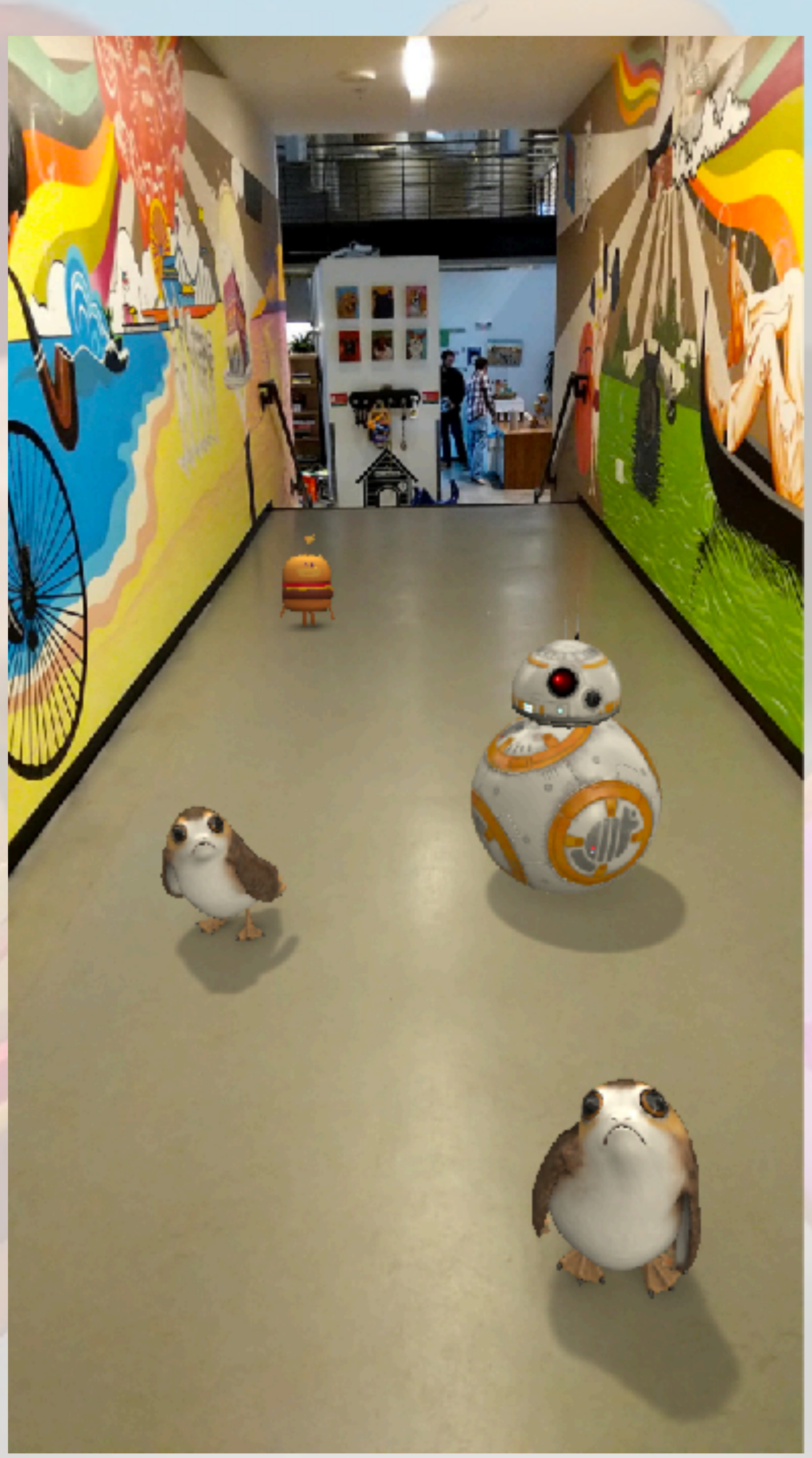
- Lots of calibration against 18% gray reference card.



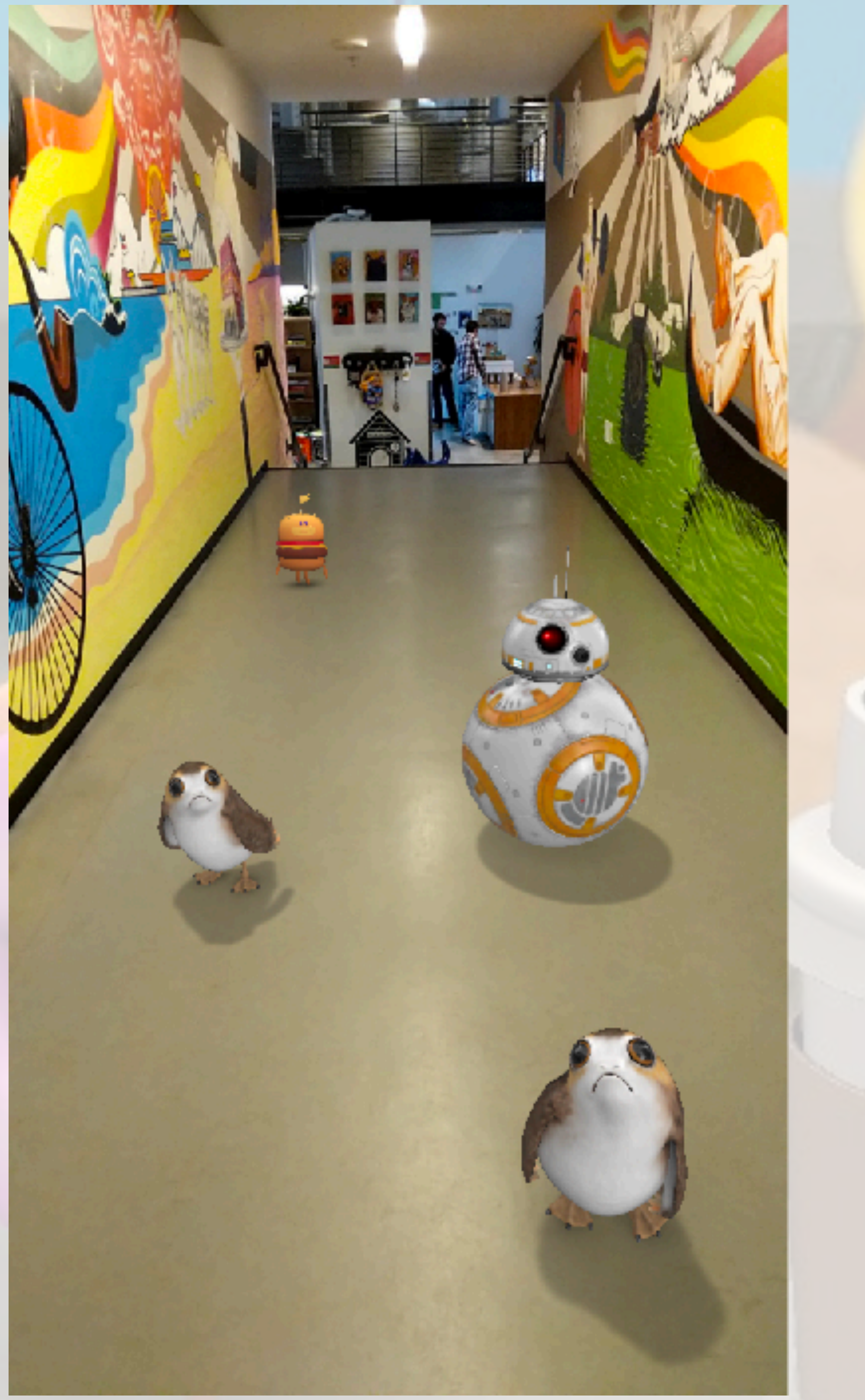
IBL: fake Image-Based Lighting



fIBL off

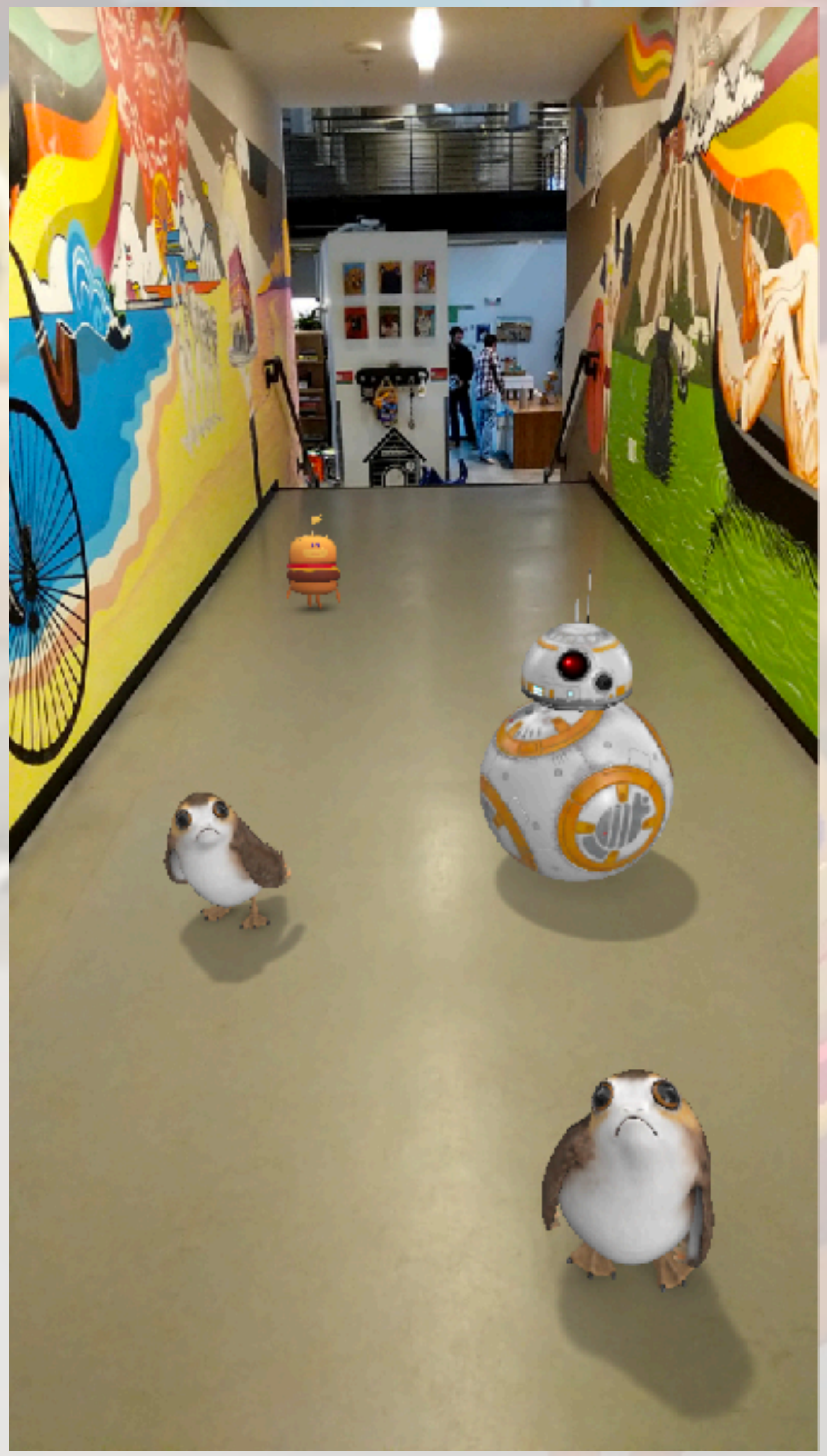


fIBL on

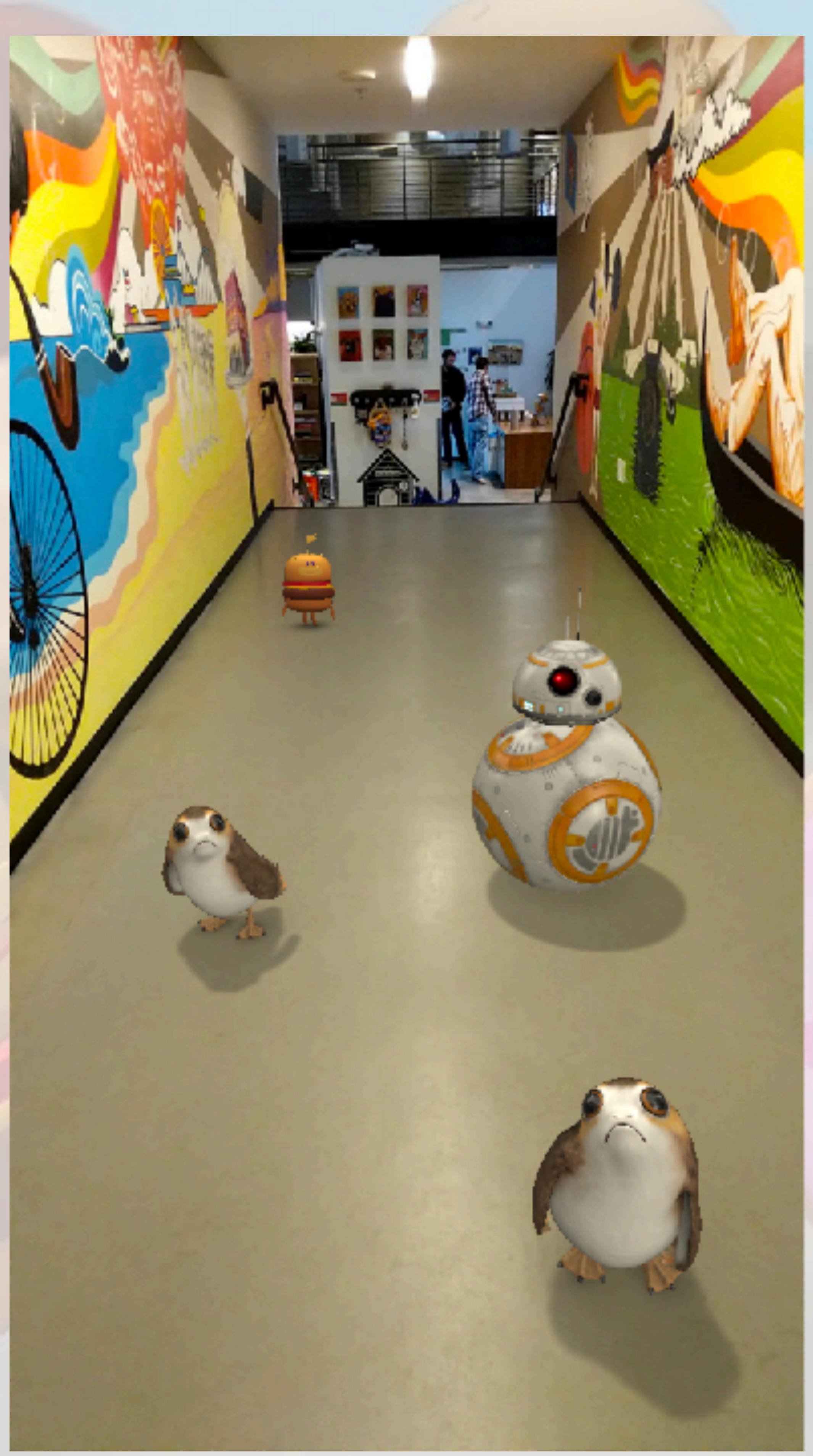


fIBL off

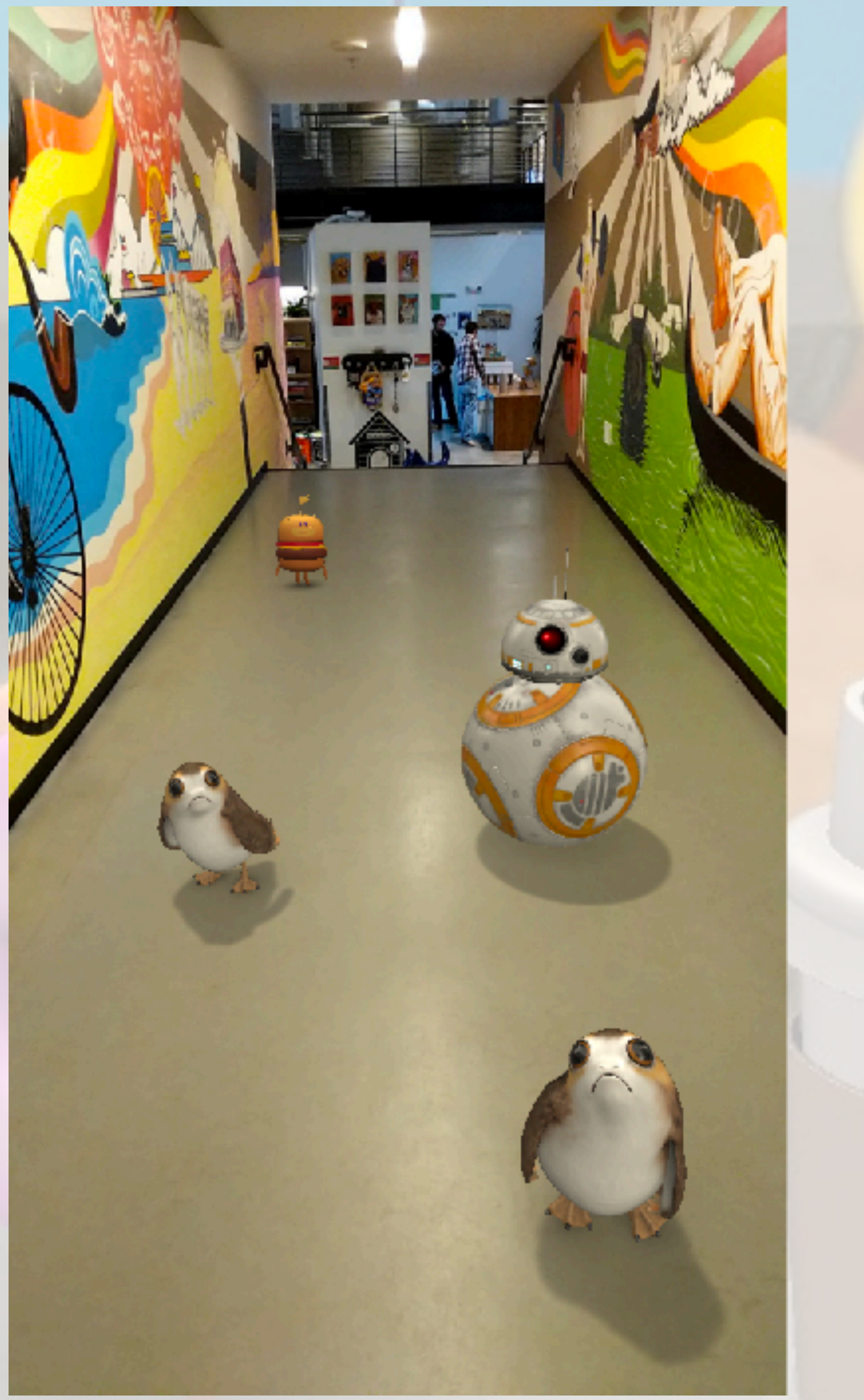
fIBL: fake Image-Based Lighting



fIBL off

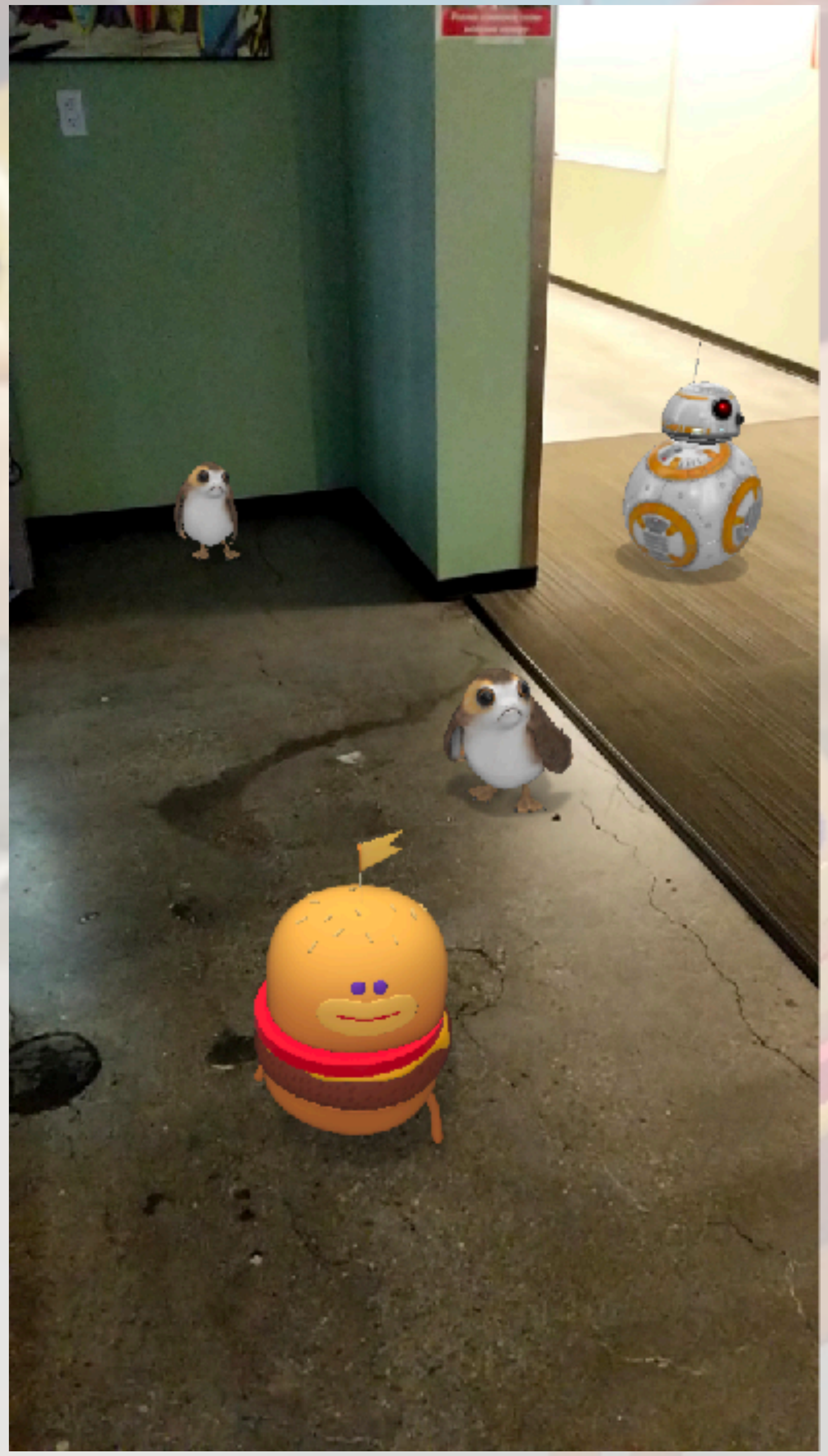


fIBL on

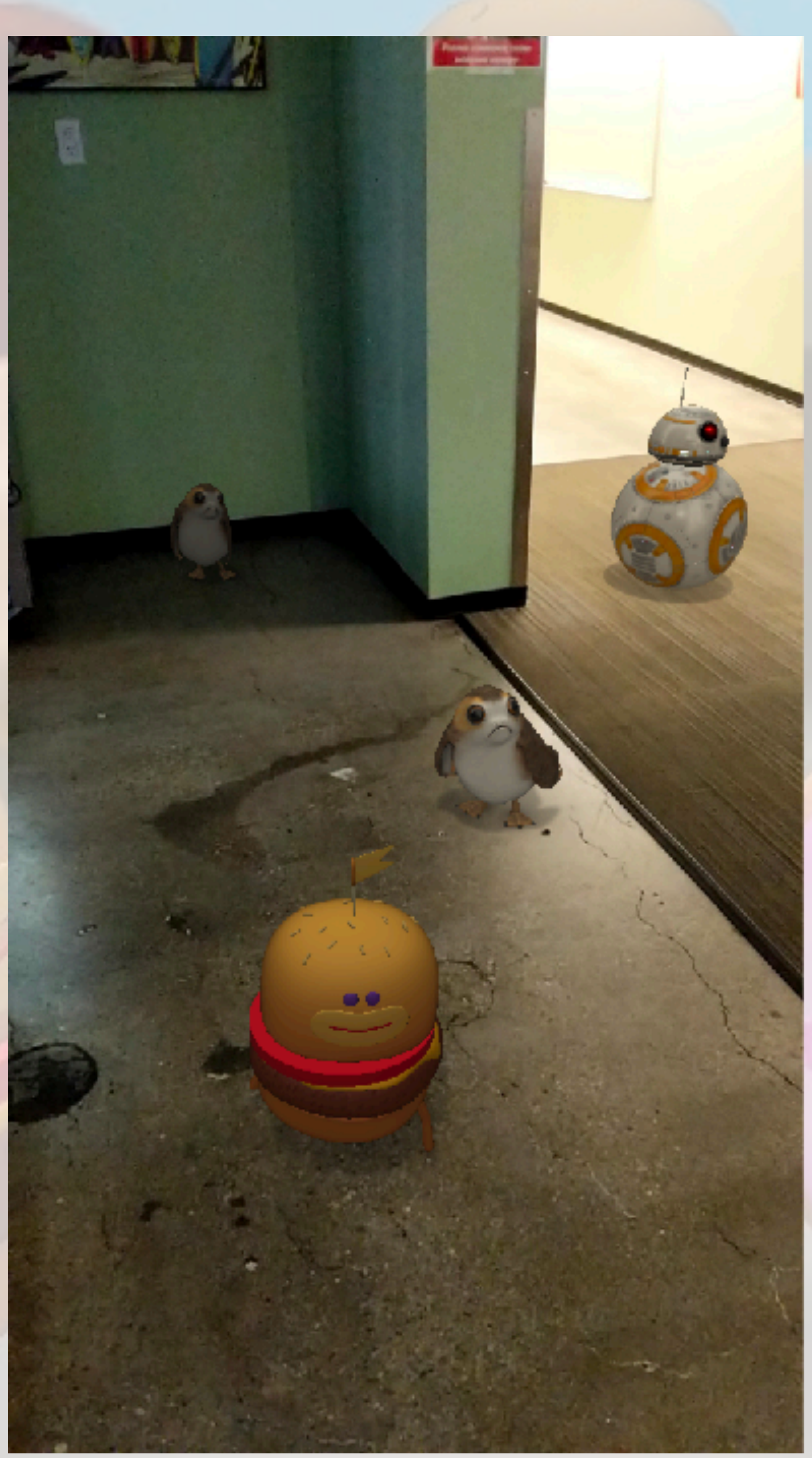


fIBL on

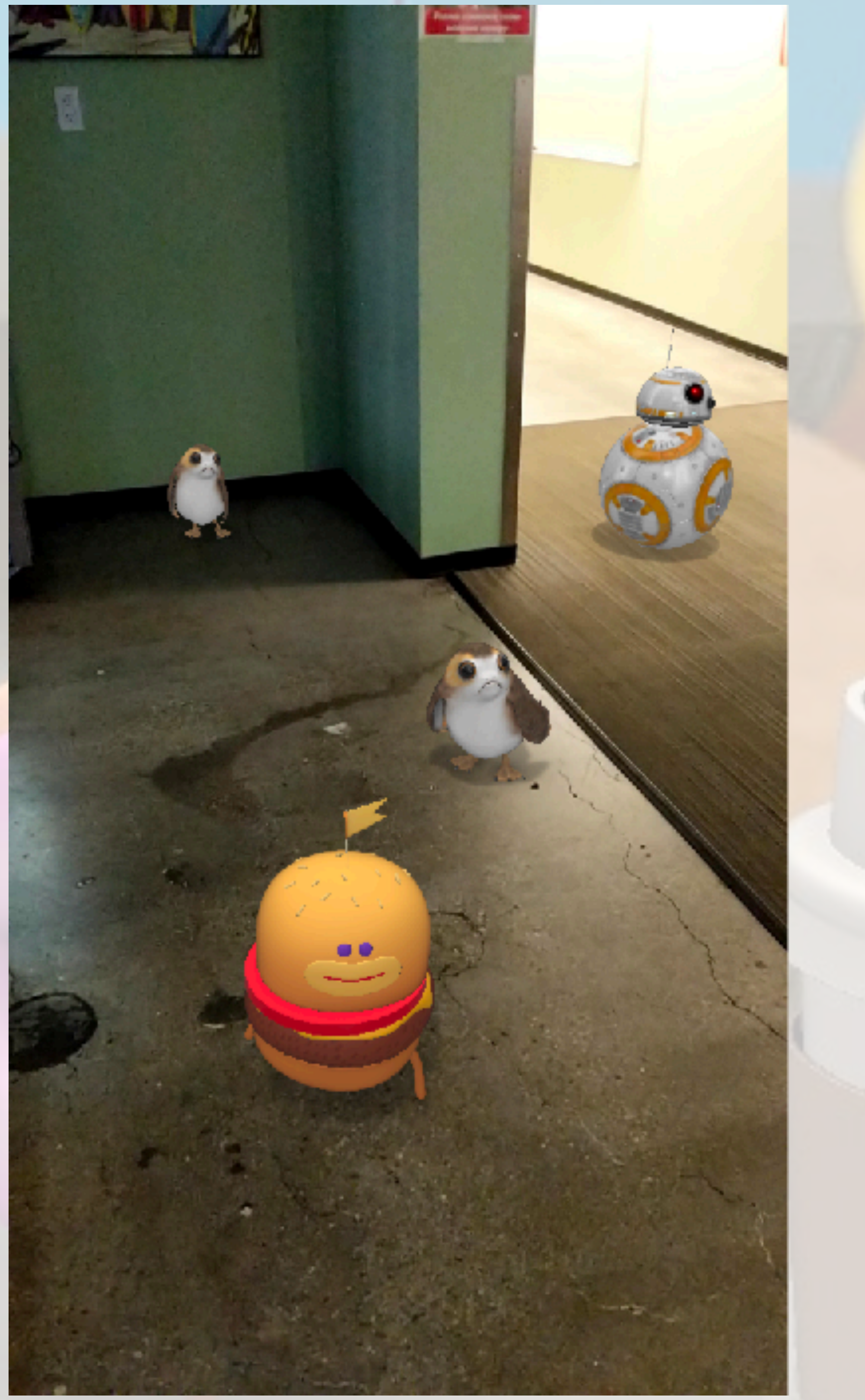
IBL: fake Image-Based Lighting



fIBL off

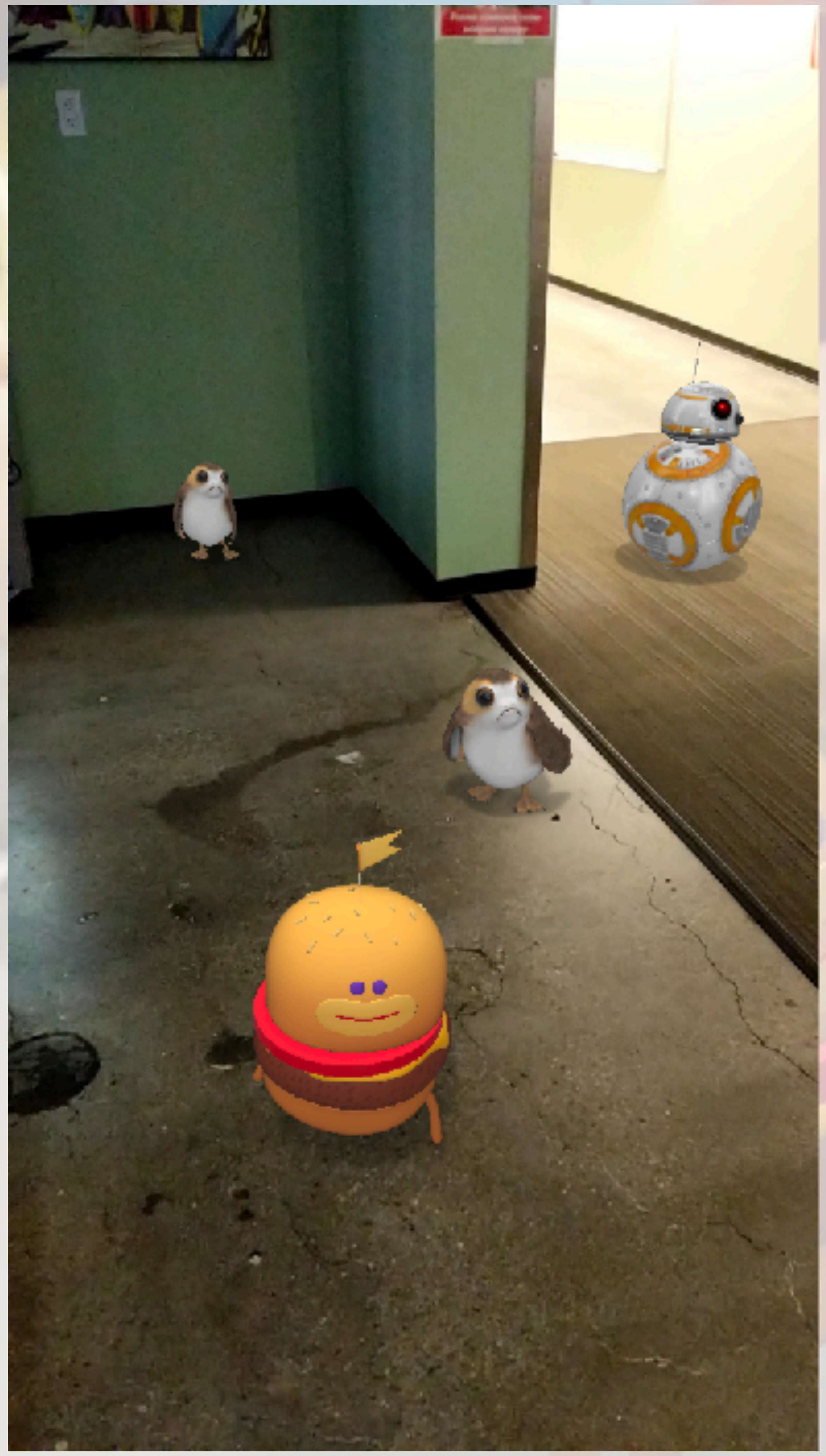


fIBL on

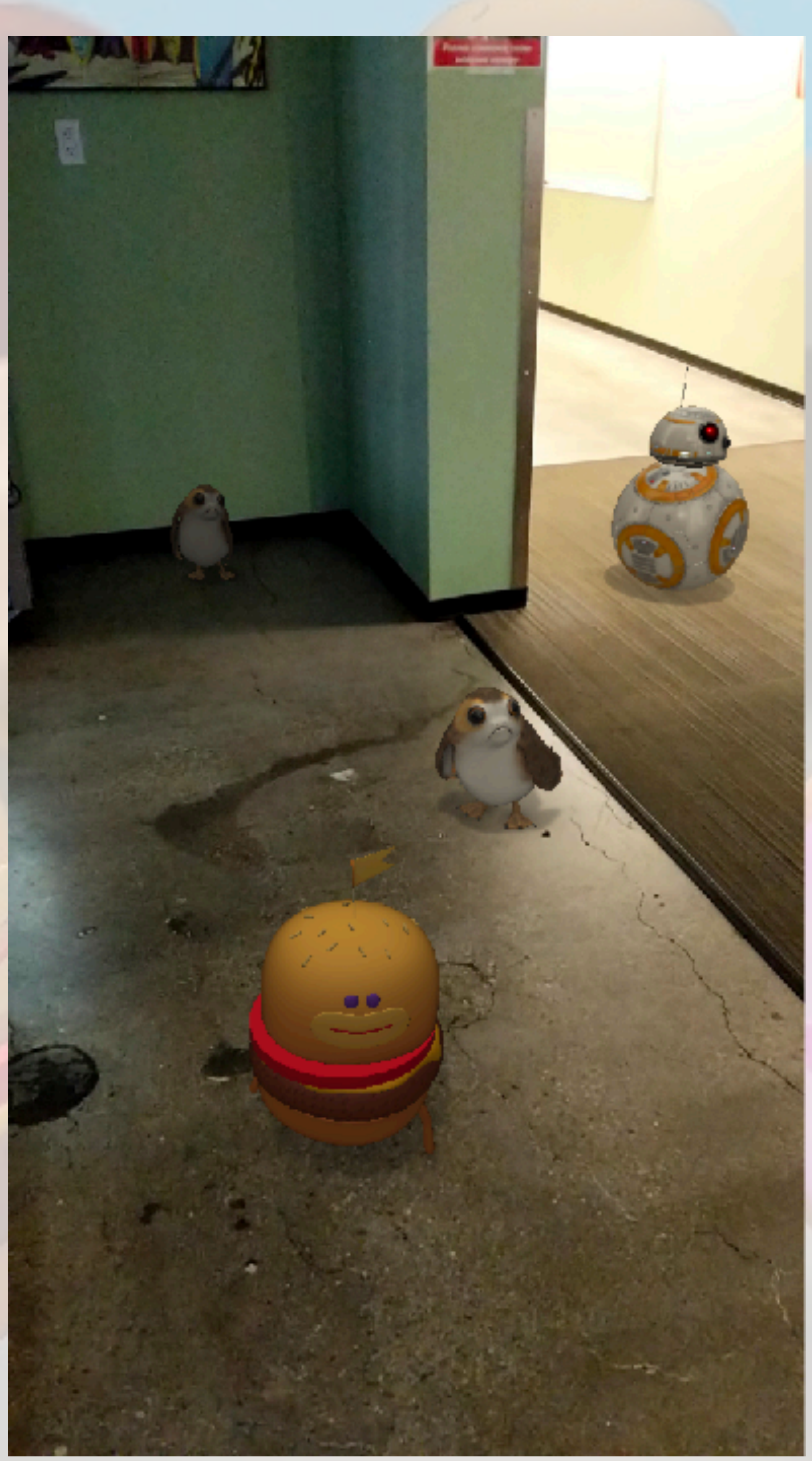


fIBL off

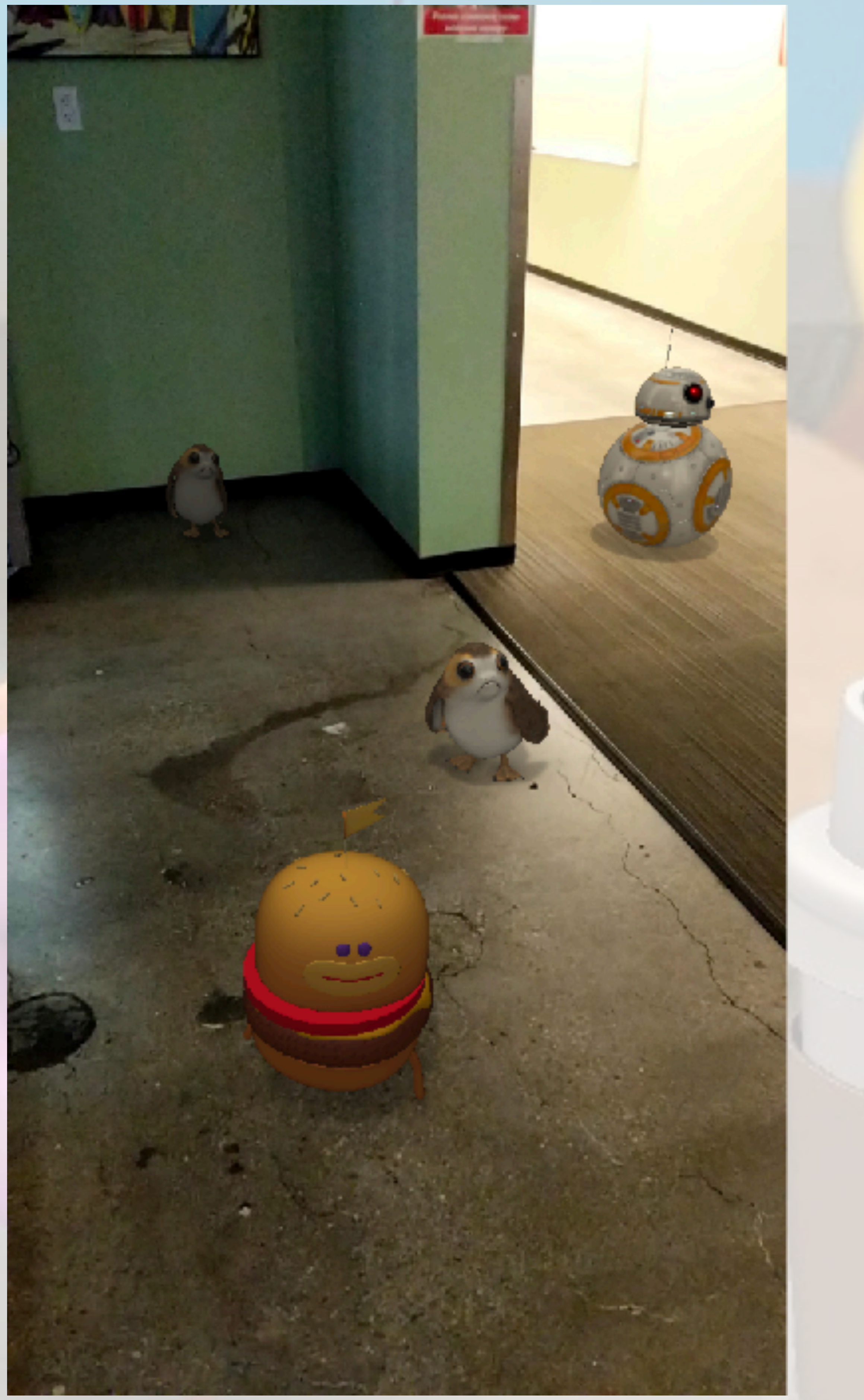
IBL: fake Image-Based Lighting



fIBL off

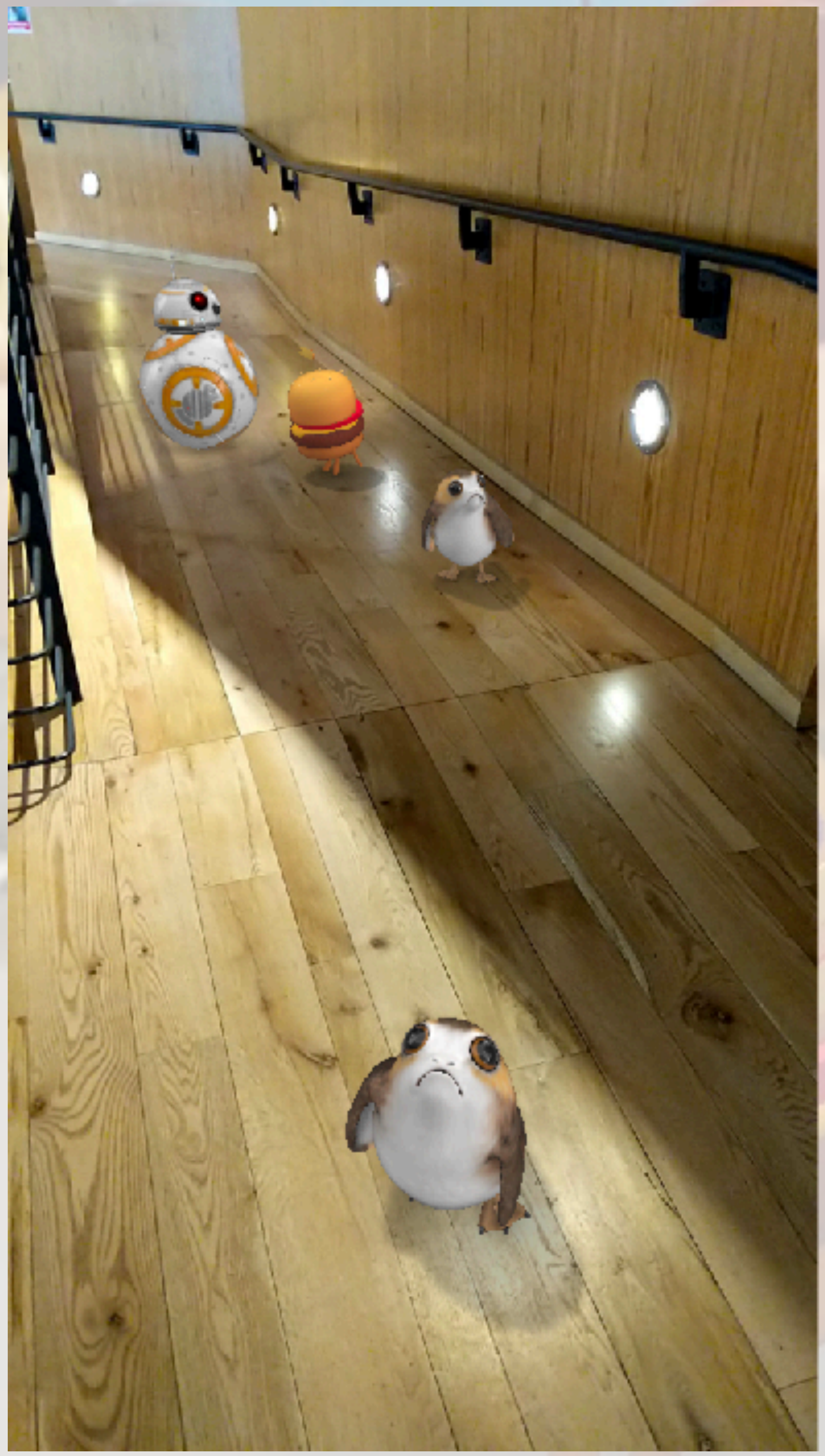


fIBL on

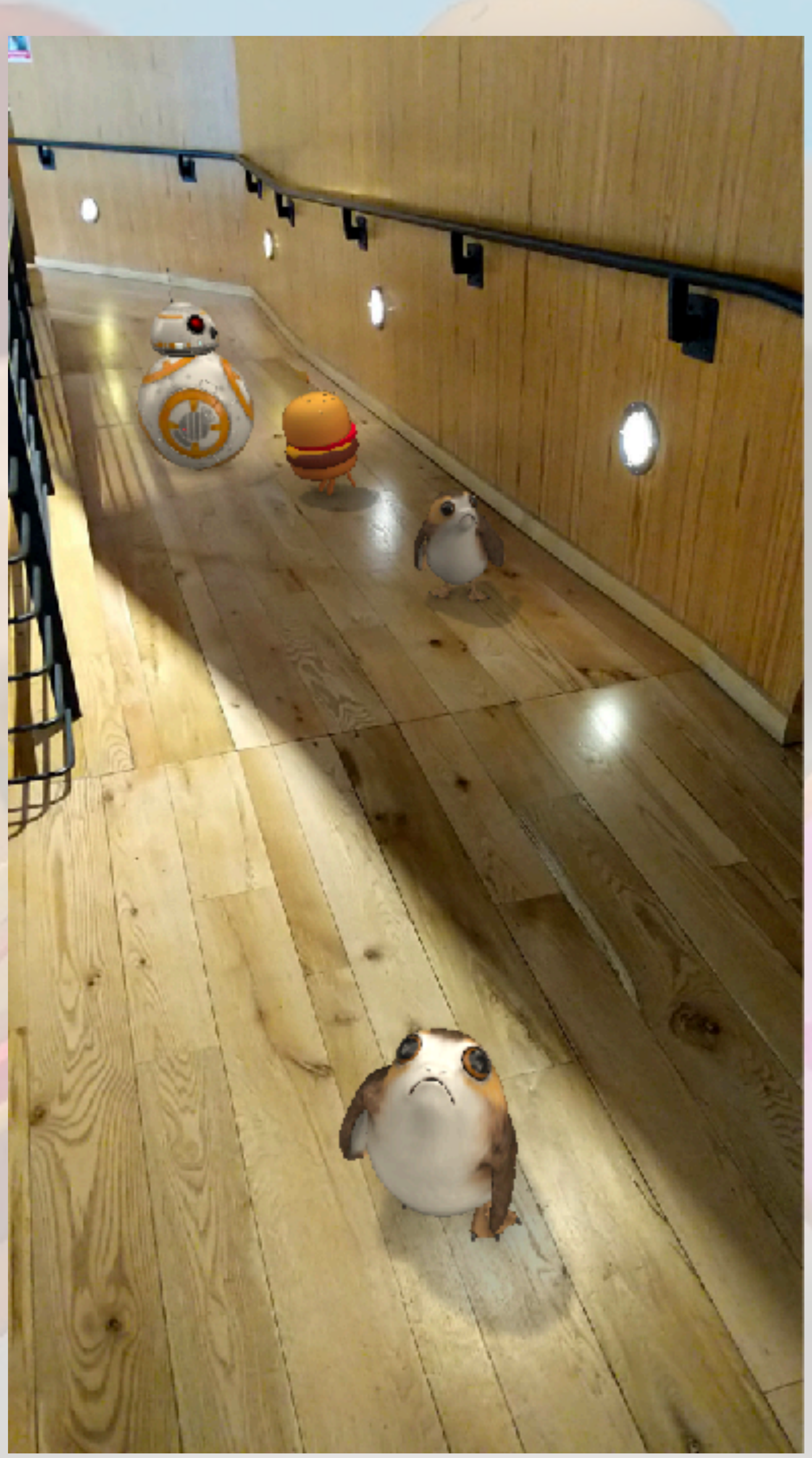


fIBL on

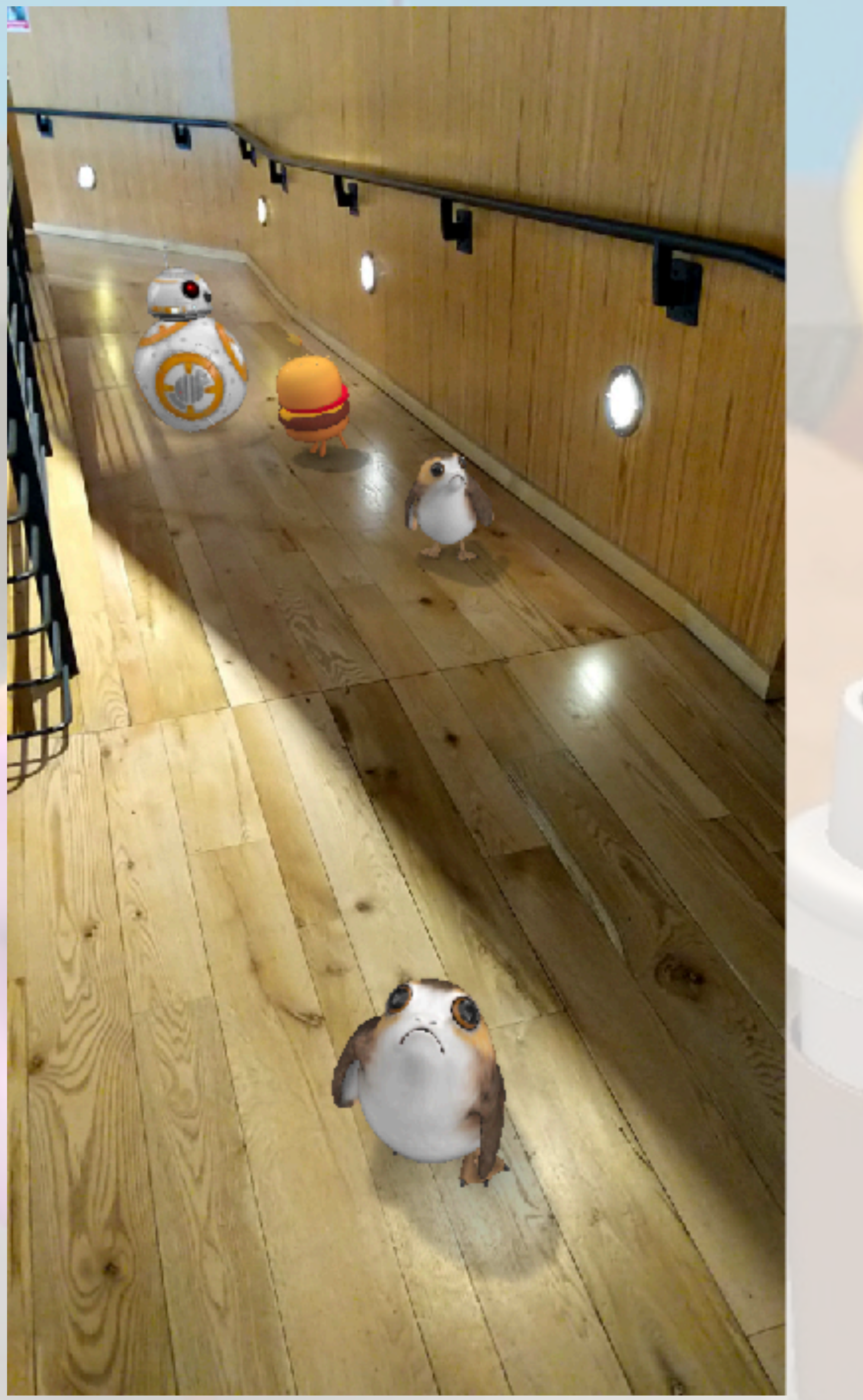
IBL: fake Image-Based Lighting



fIBL off

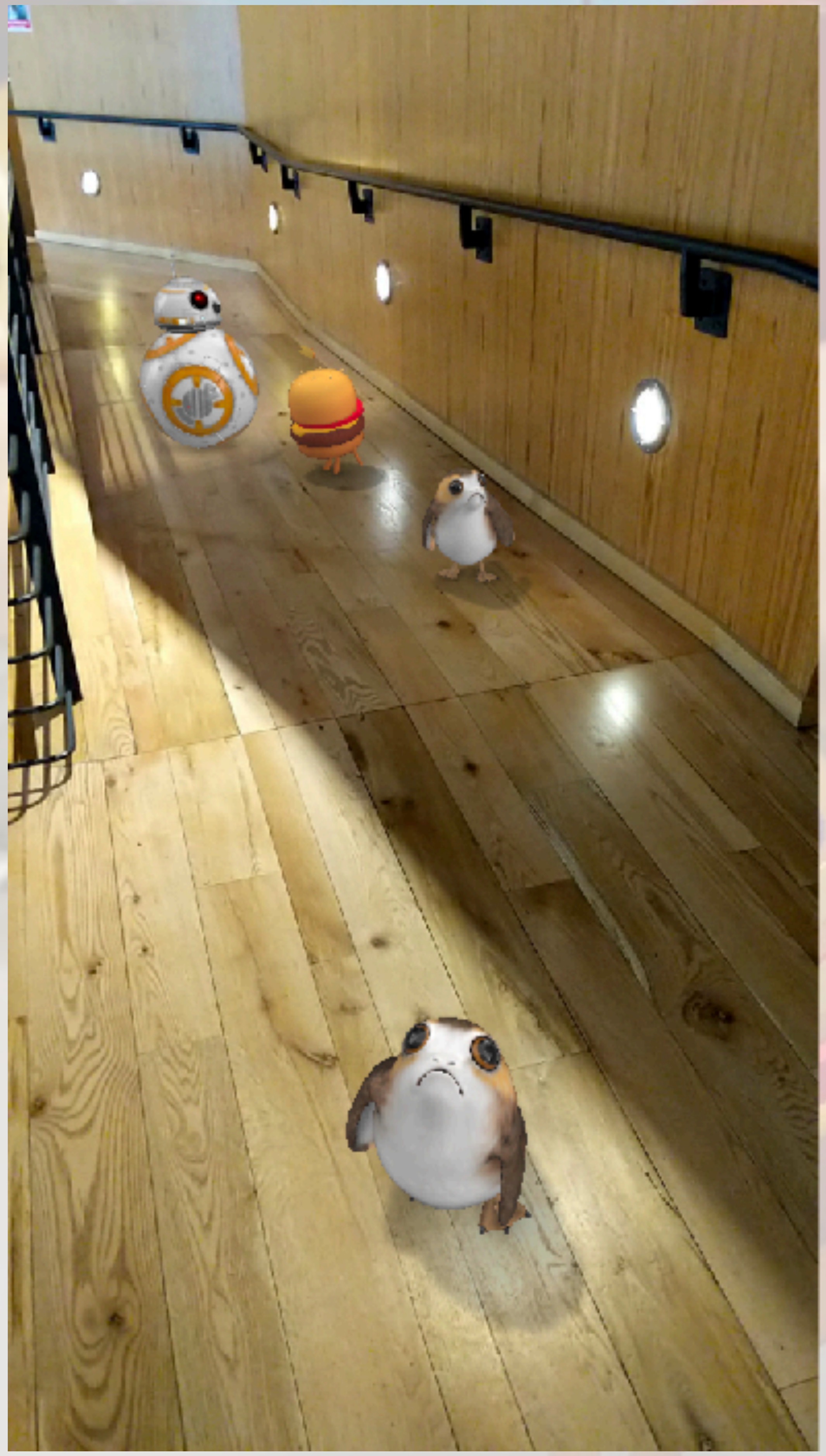


fIBL on

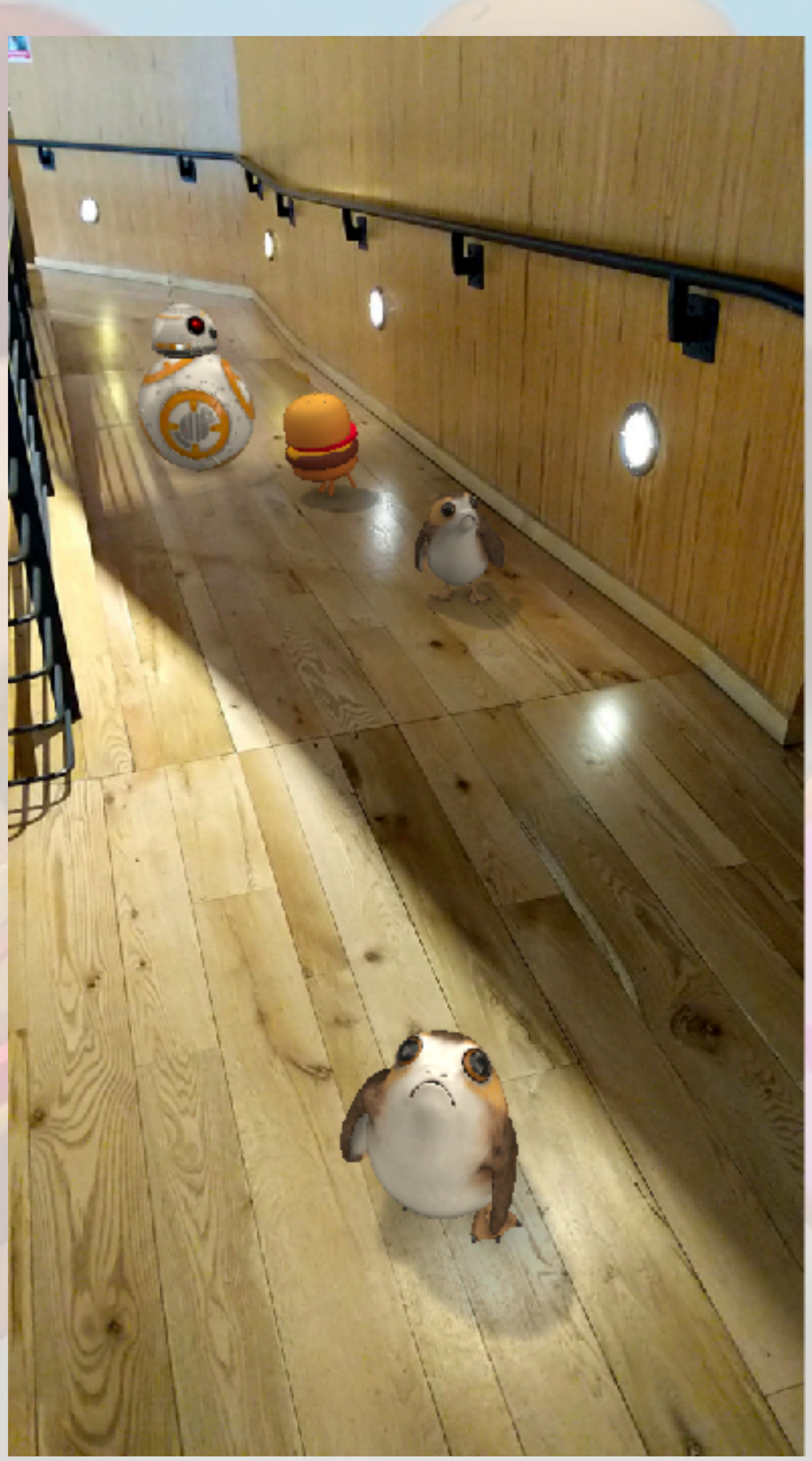


fIBL off

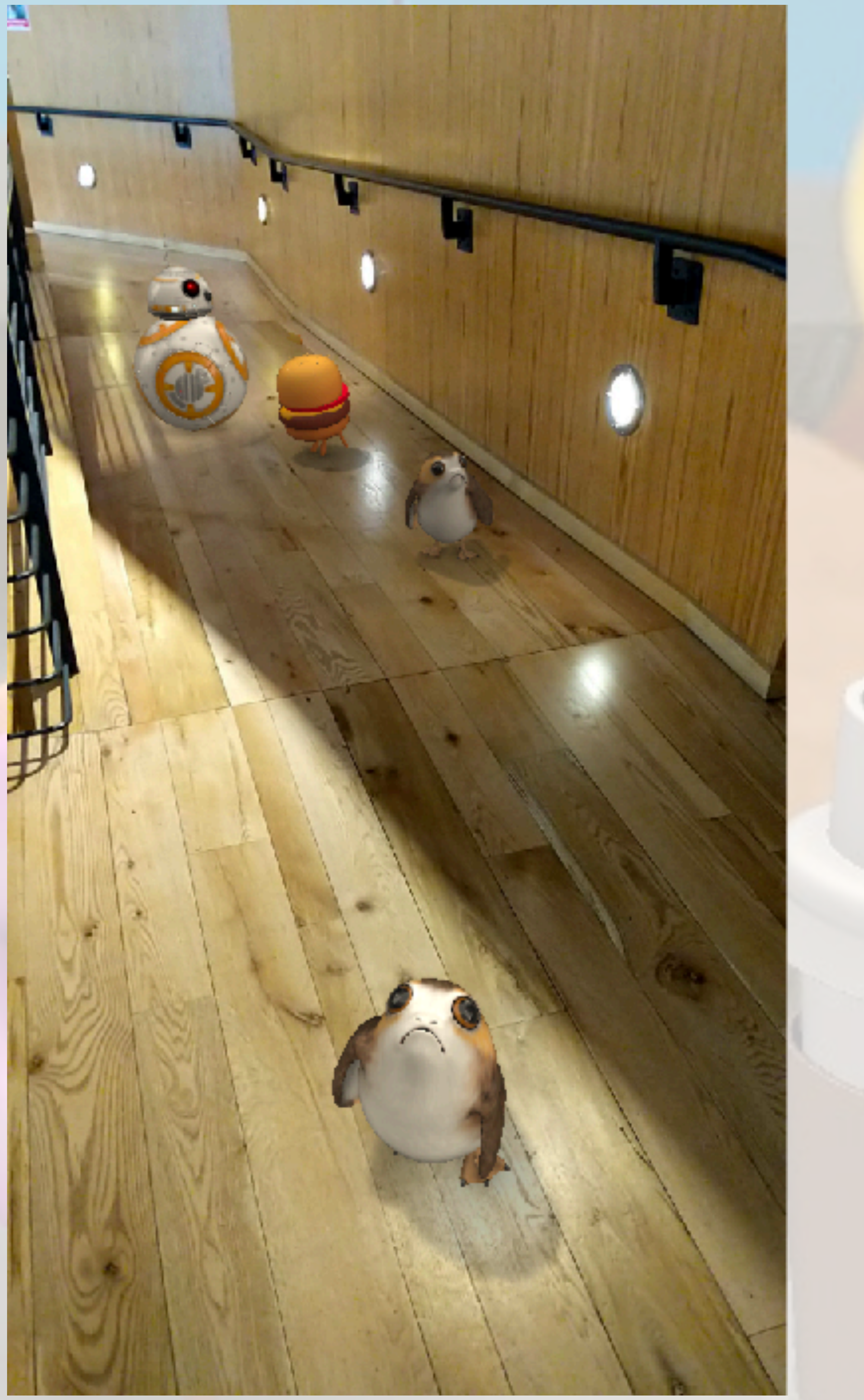
IBL: fake Image-Based Lighting



fIBL off

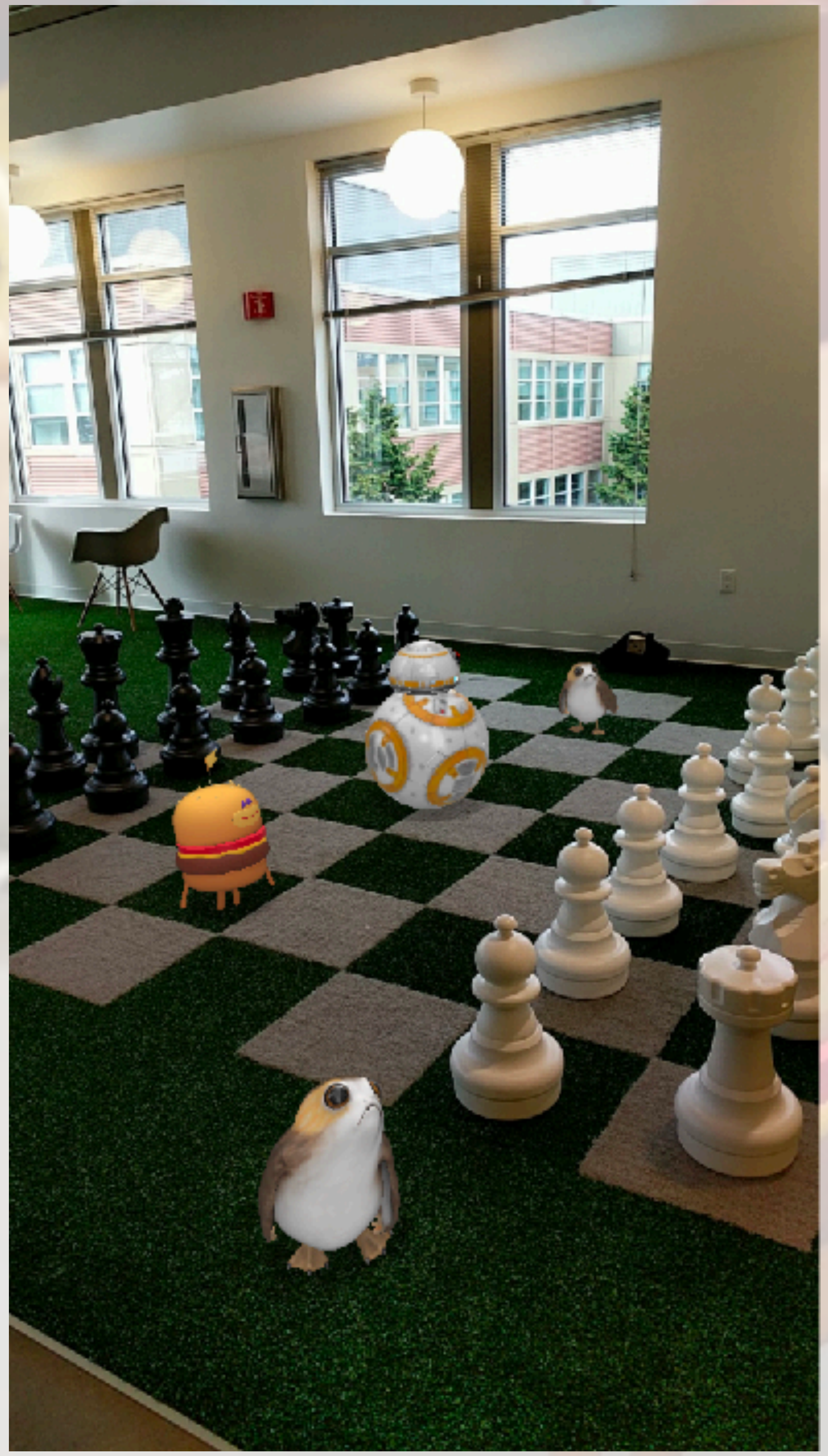


fIBL on

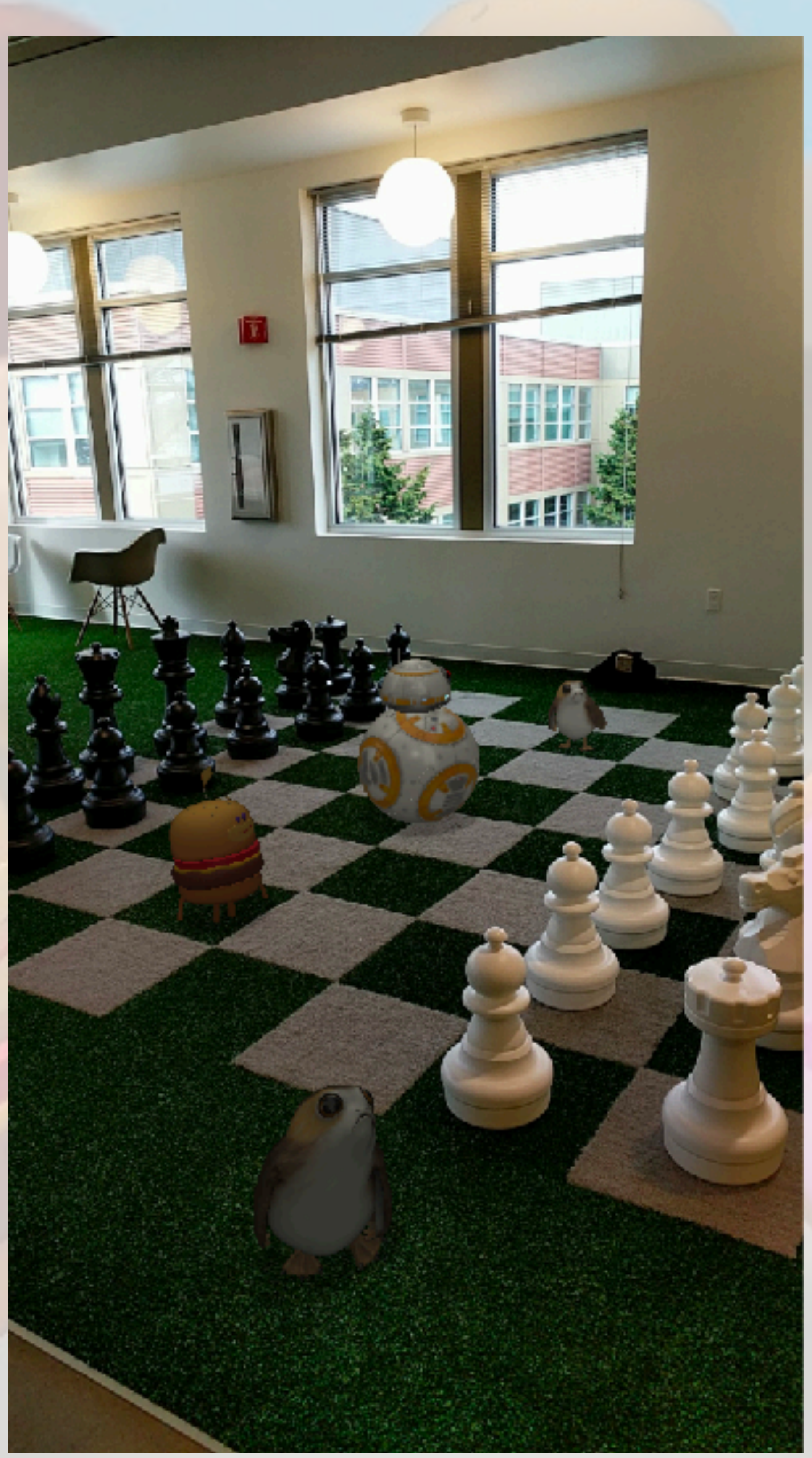


fIBL on

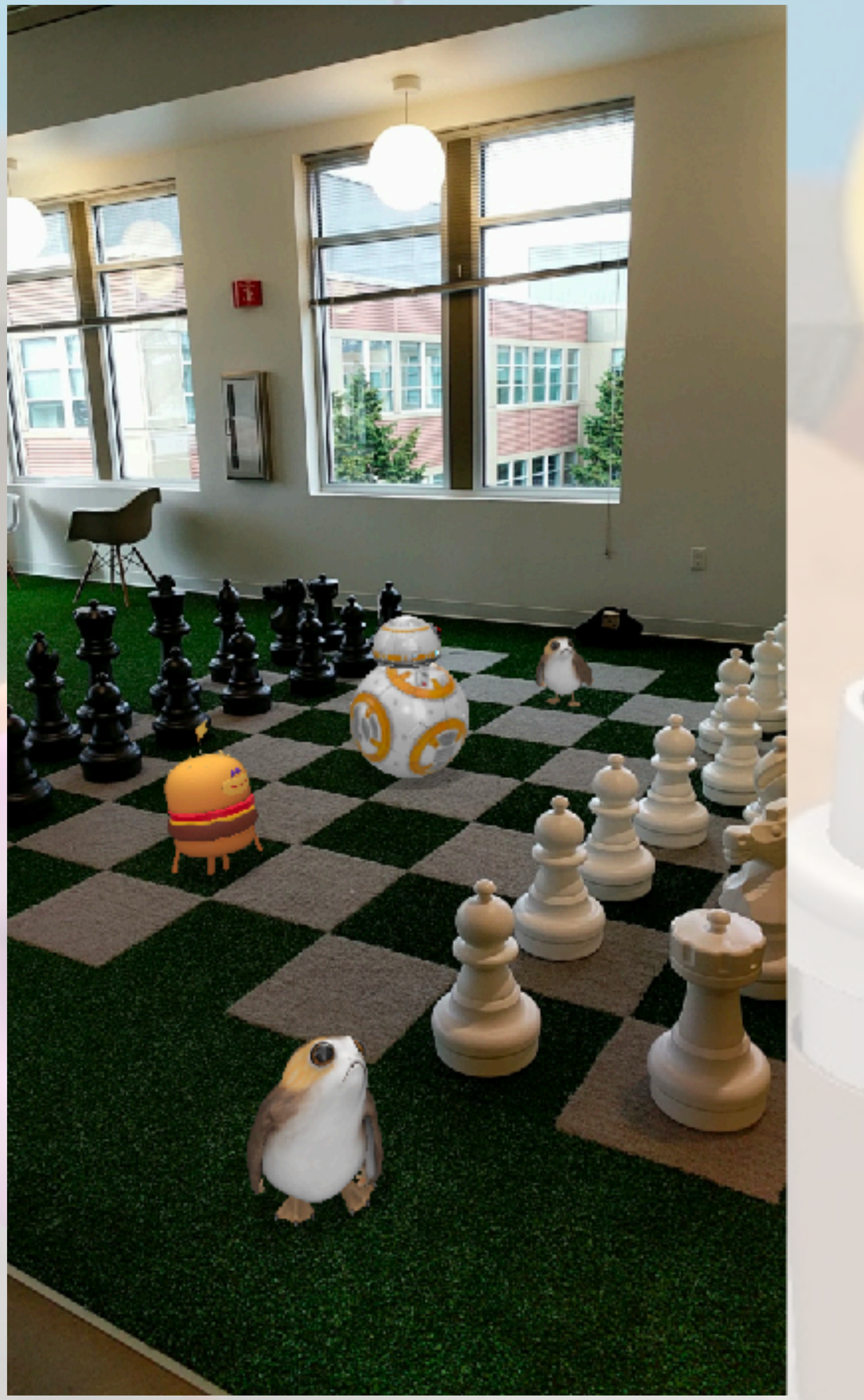
IBL: fake Image-Based Lighting



fIBL off

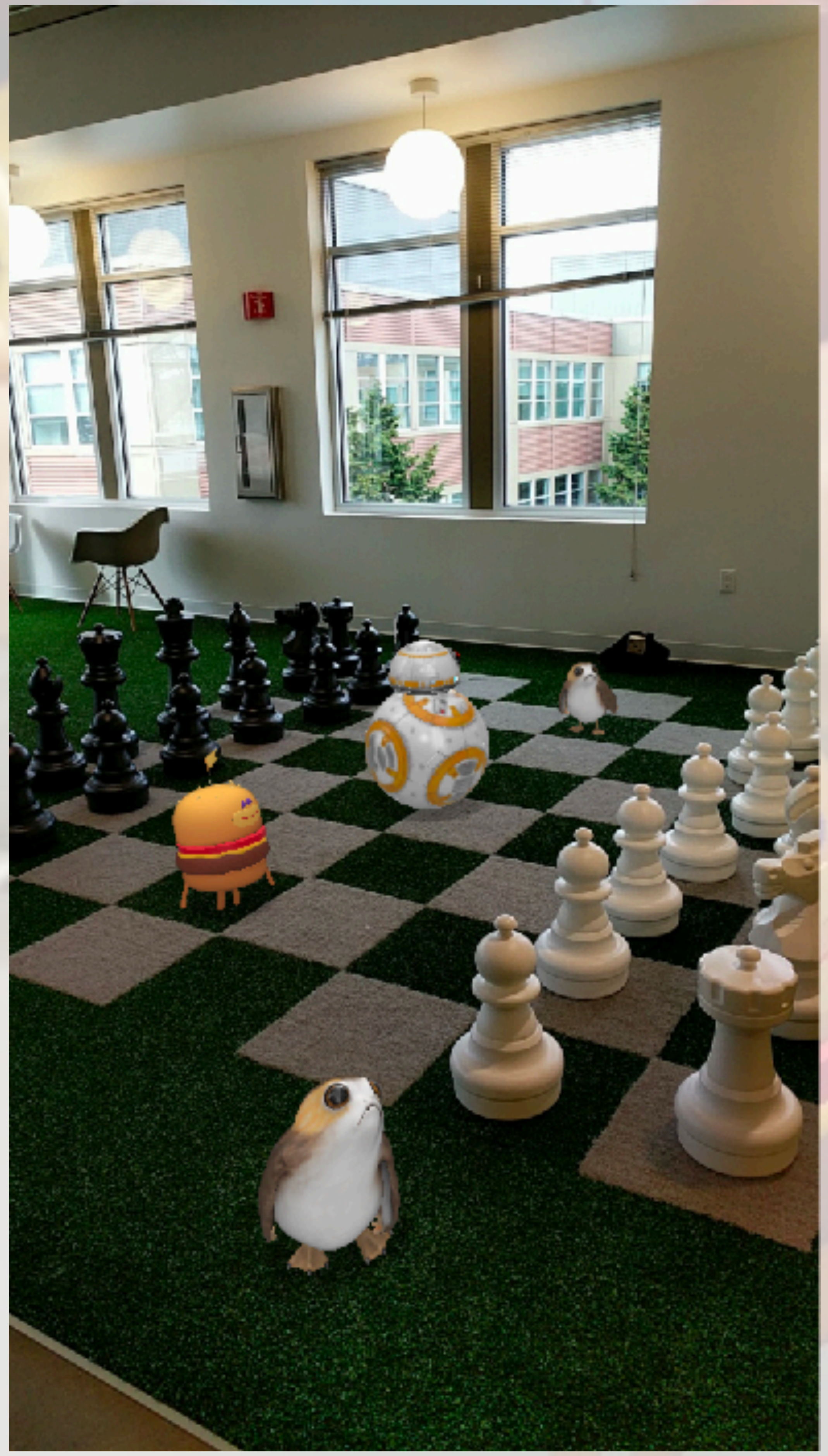


fIBL on

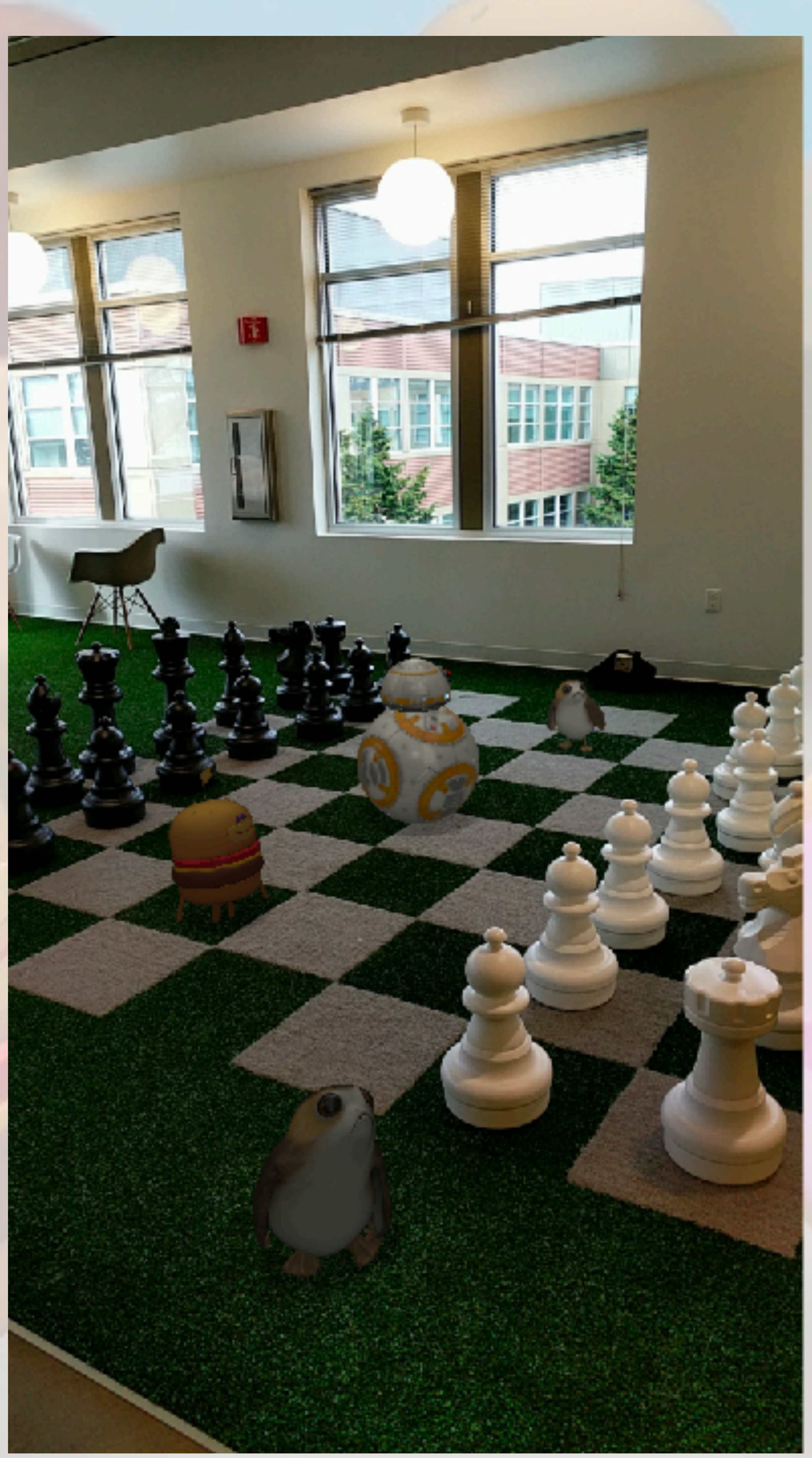


fIBL off

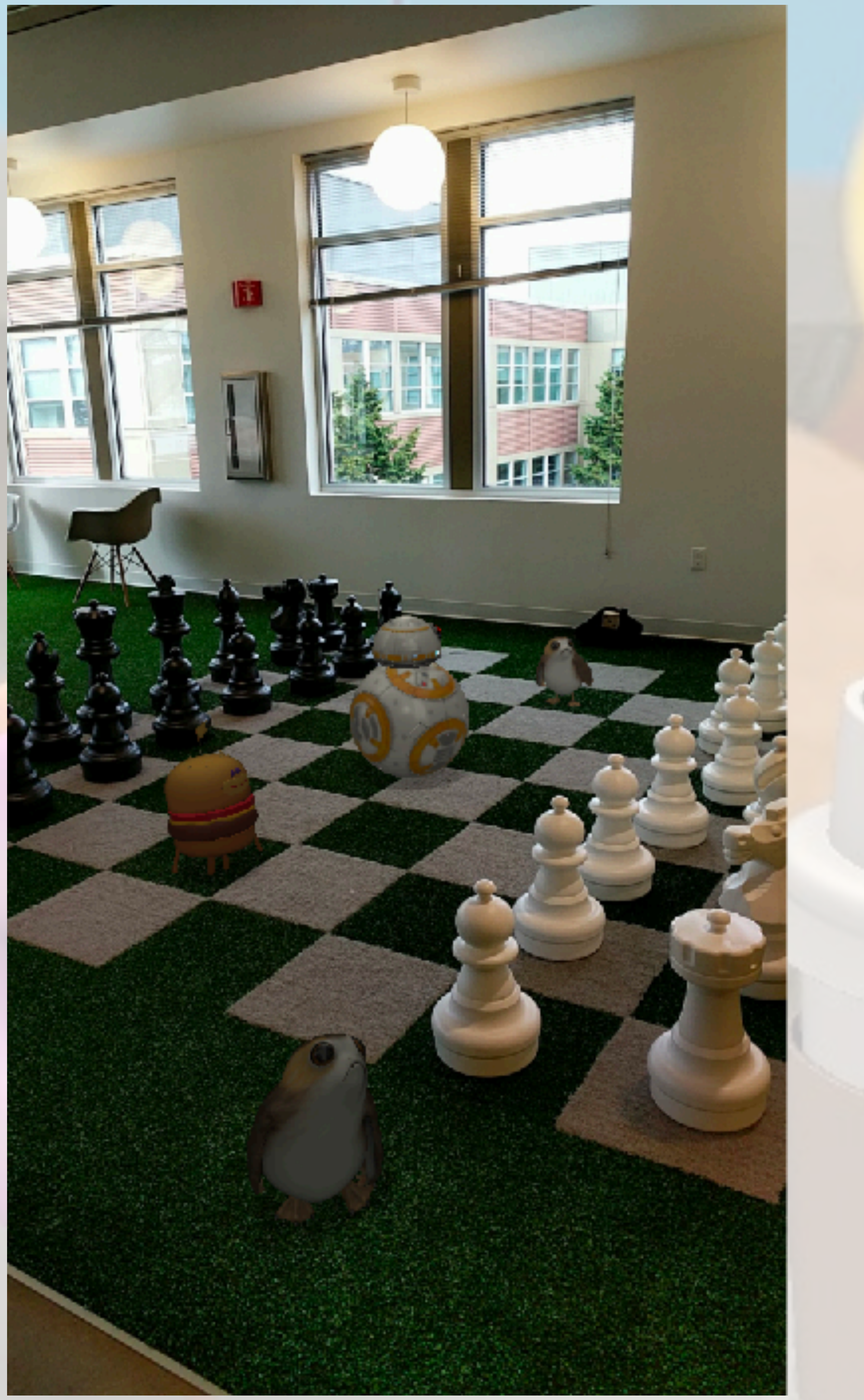
IBL: fake Image-Based Lighting



fIBL off



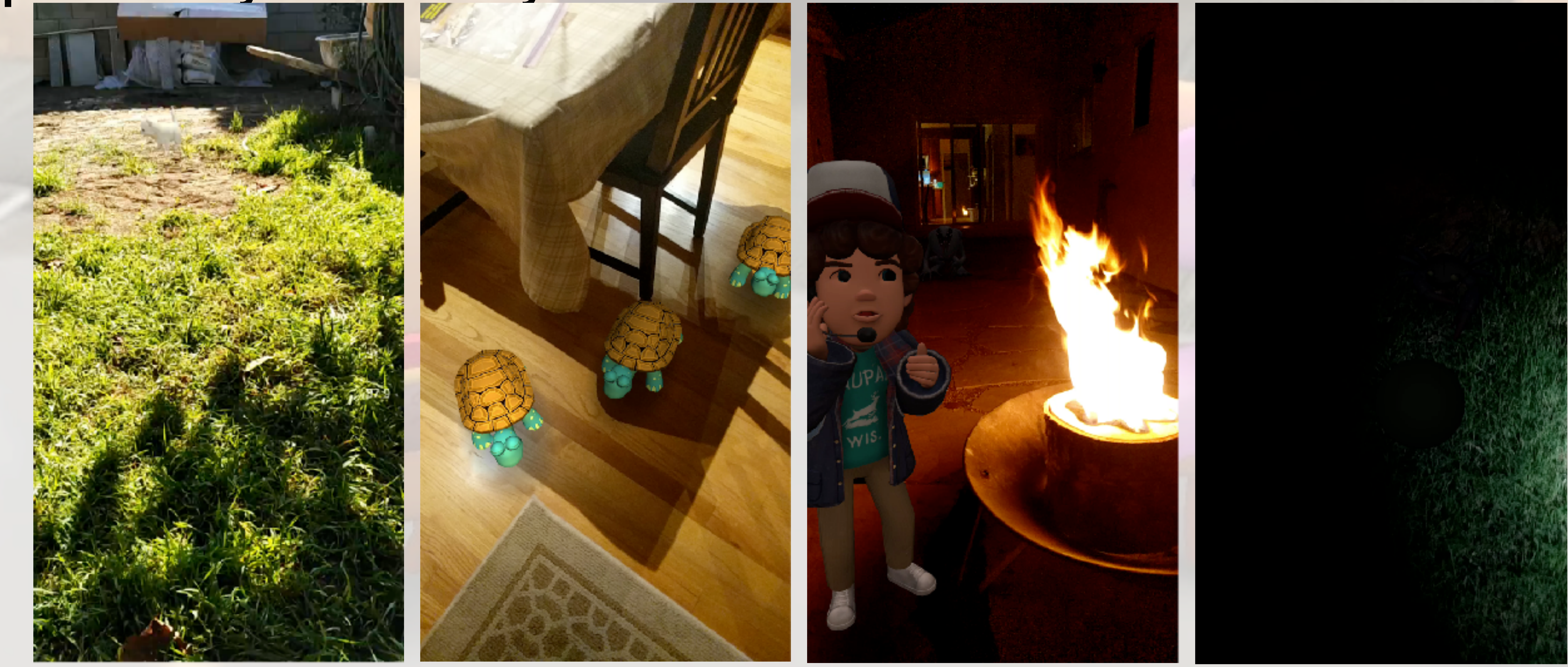
fIBL on



fIBL on

fIBL: fake Image-Based Lighting

- fIBL existed in prototype, but ran slowly
  - Updated only while dragging.
- Final app moved execution to GPU
  - Responds dynamically for all stickers at all frames.



Camera-Based Specular Reflections

Goal: Incorporate camera feed detail into specular reflections.

- Enhances glossy surfaces.
- We already have camera feed texture.

Problem: Phone camera has

- Limited dynamic range.
- Very narrow field of view.
  - Unknown incident light for most reflections.

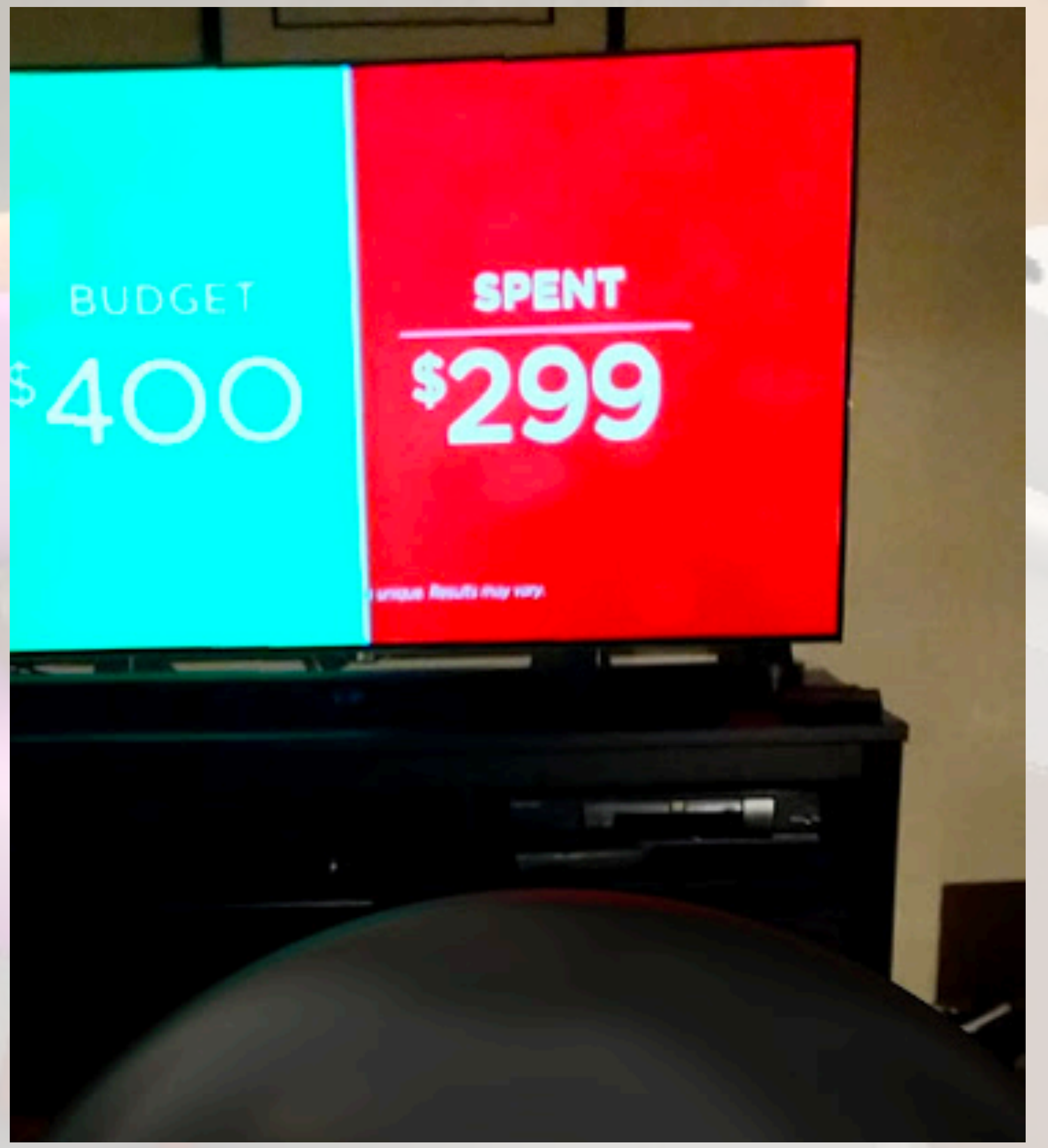
Camera-Based Specular Reflections

Solution: Incorporate camera feed *only* where plausible.

- At grazing angles, where incident rays are in camera view.

Hence:

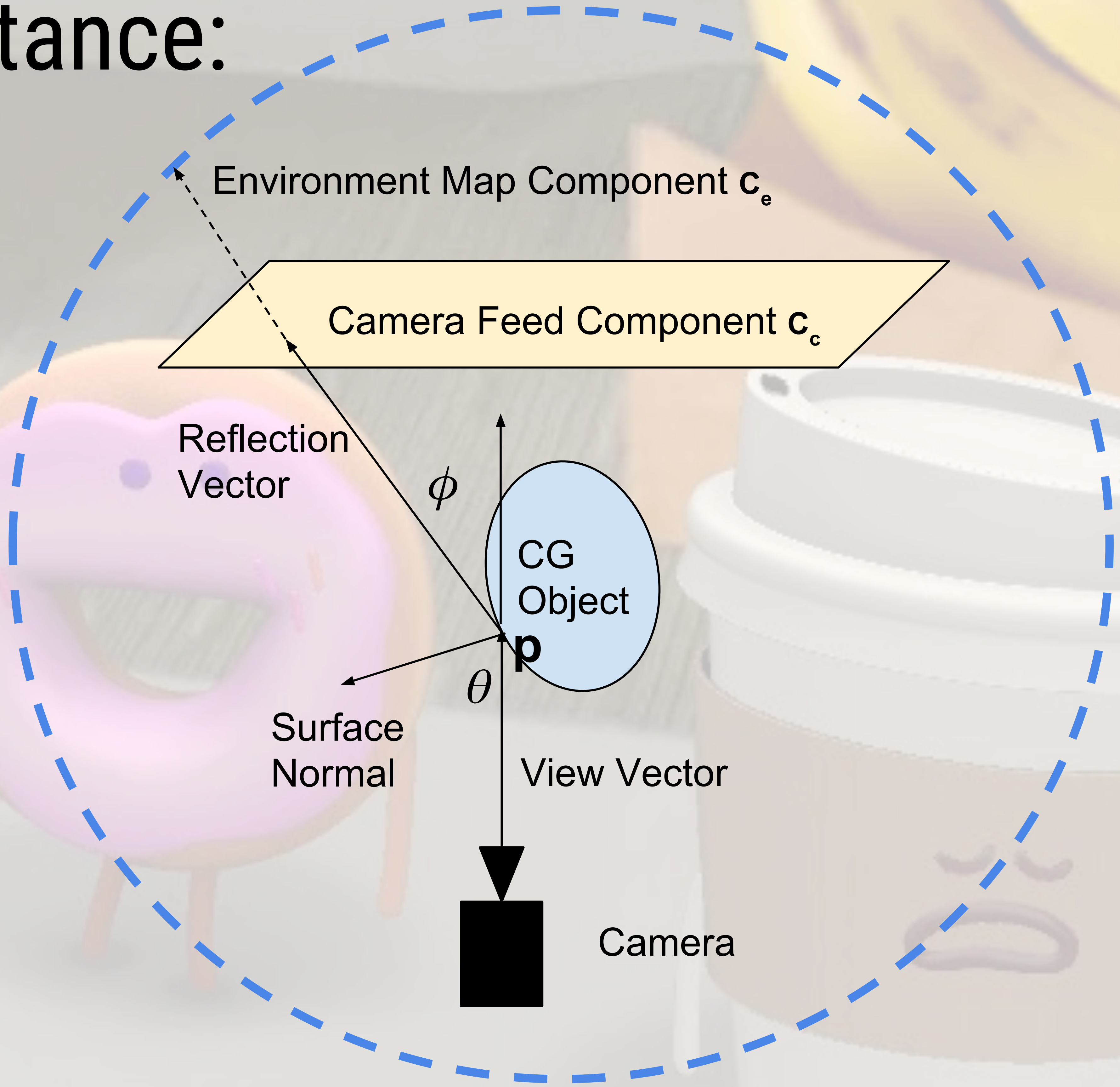
- Use synthetic cube map as default.
- Blend in Screen-Space Reflections at grazing angles w/ sharp falloff.
- Accounts for the blue/red color along sphere silhouette .



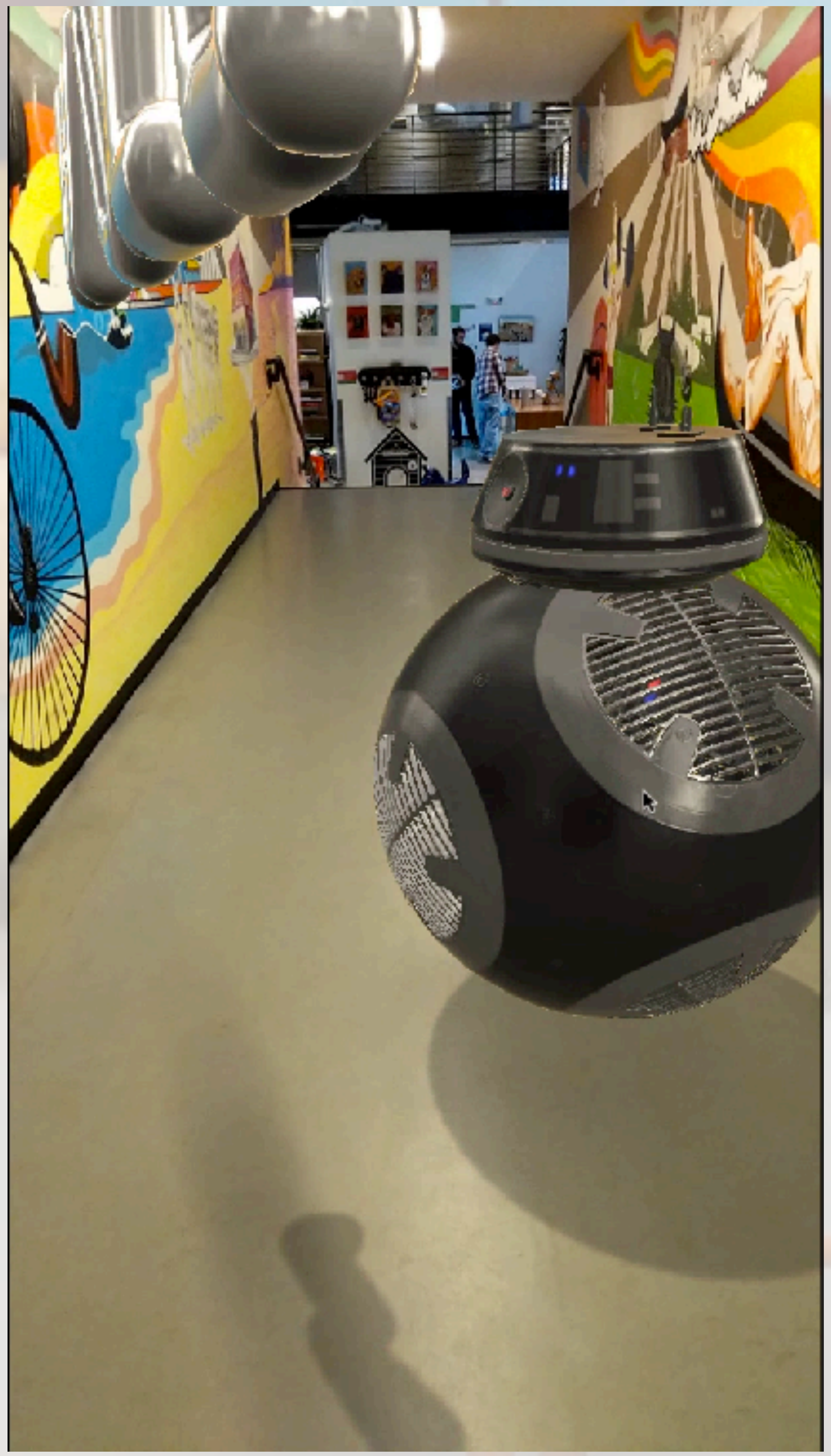
Camera-Based Specular Reflections

Our approach:

- Blend cube map to camera feed using $(1 - \cos\theta)^5$
  - Schlick’s approximation to Fresnel reflectance:
- Adjust camera feed brightness to match synthetic Environment Map.
  - Avoid reflections glowing brighter than camera pixels (tone mapping)
- Blur camera feed based on BRDF
  - No prefiltering of camera feed, so mipmap levels differ.



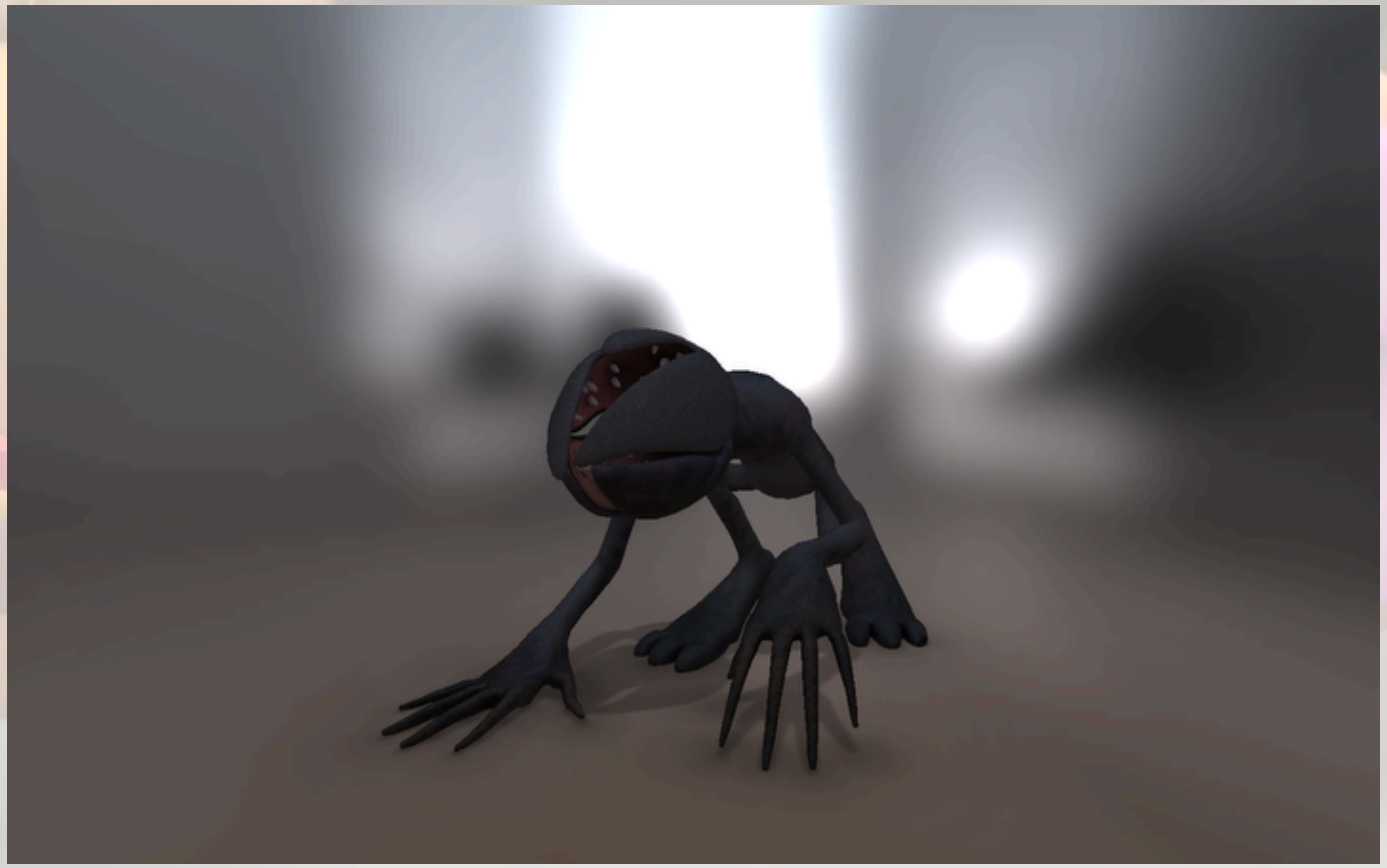
Camera-Based Specular Reflections



Shadows

Critical in AR because they

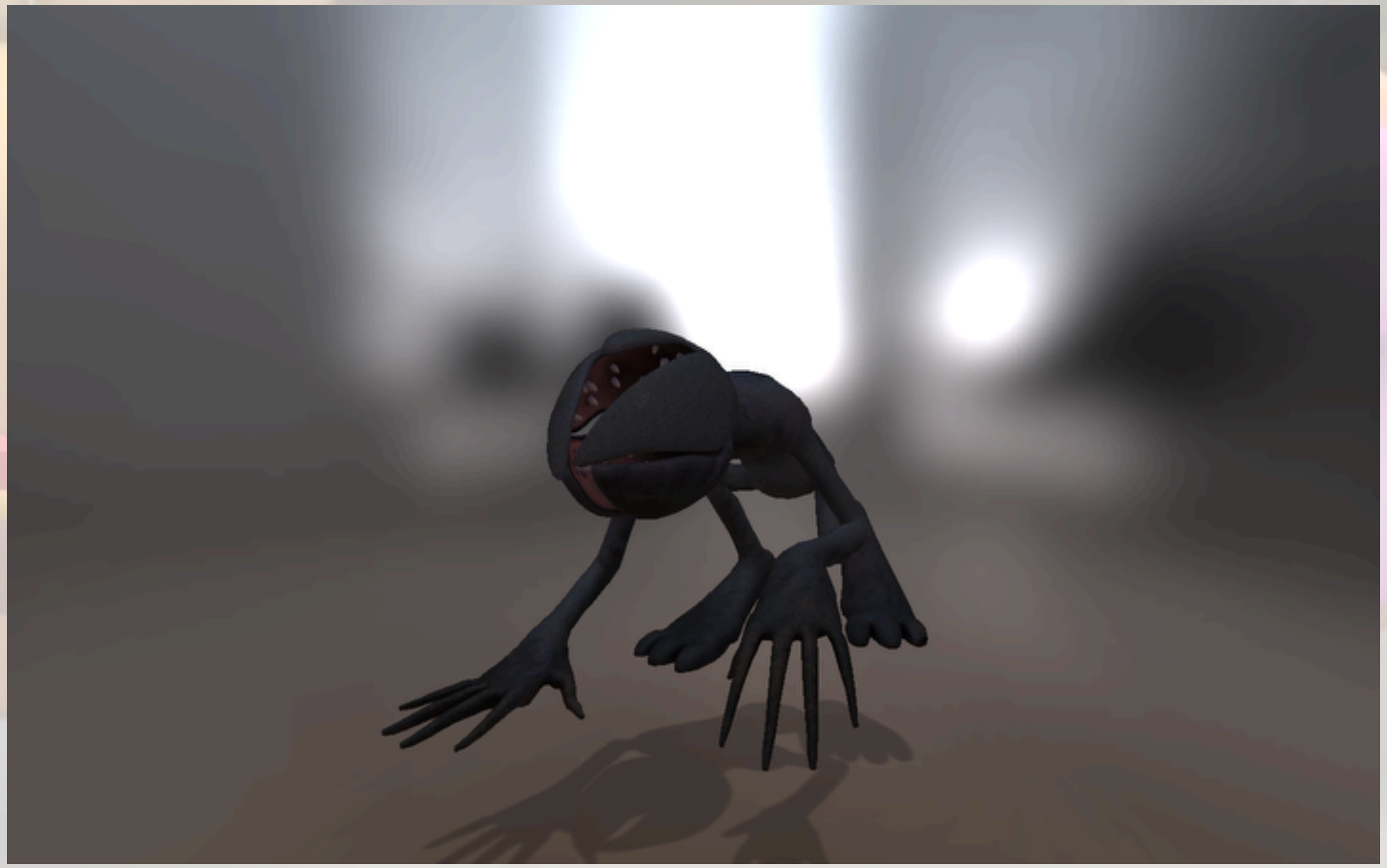
- ground stickers to the real floor (avoids “floating”).
- provide a powerful hint as to sticker height.
- disambiguate between altitude and depth.



Shadows

Critical in AR because they

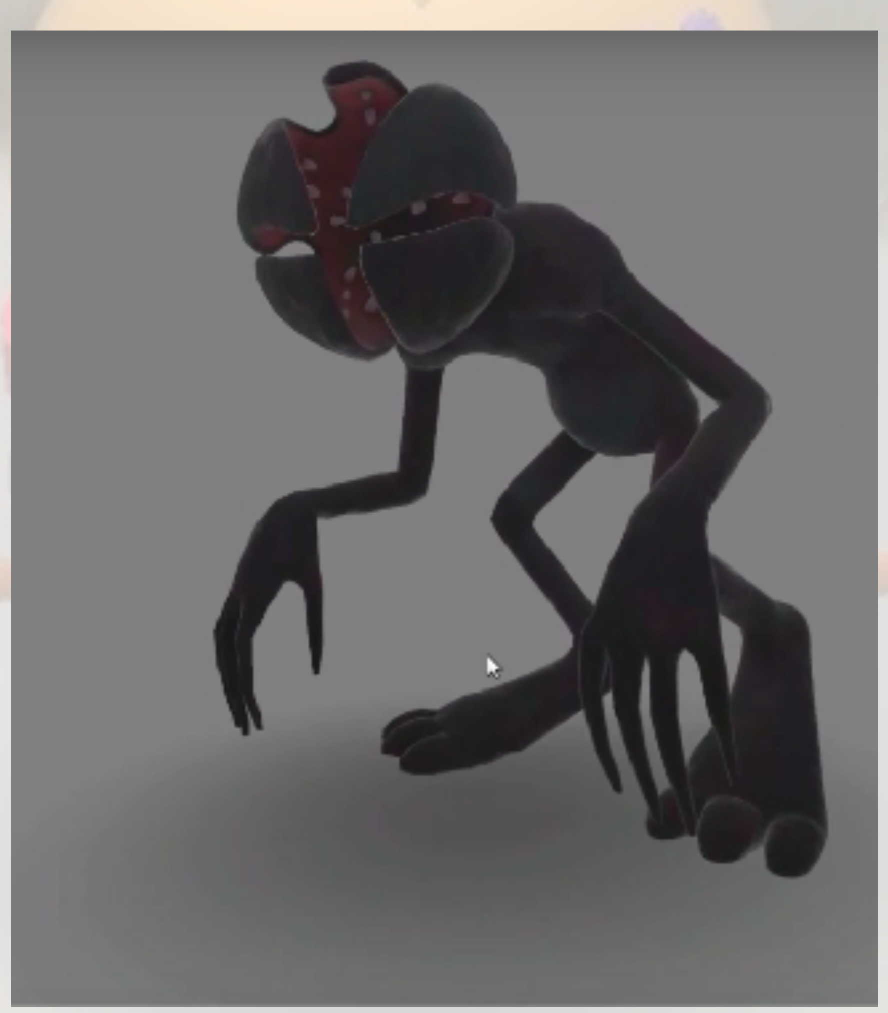
- ground stickers to the real floor (avoids “floating”).
- provide a powerful hint as to sticker height.
- disambiguate between altitude and depth.

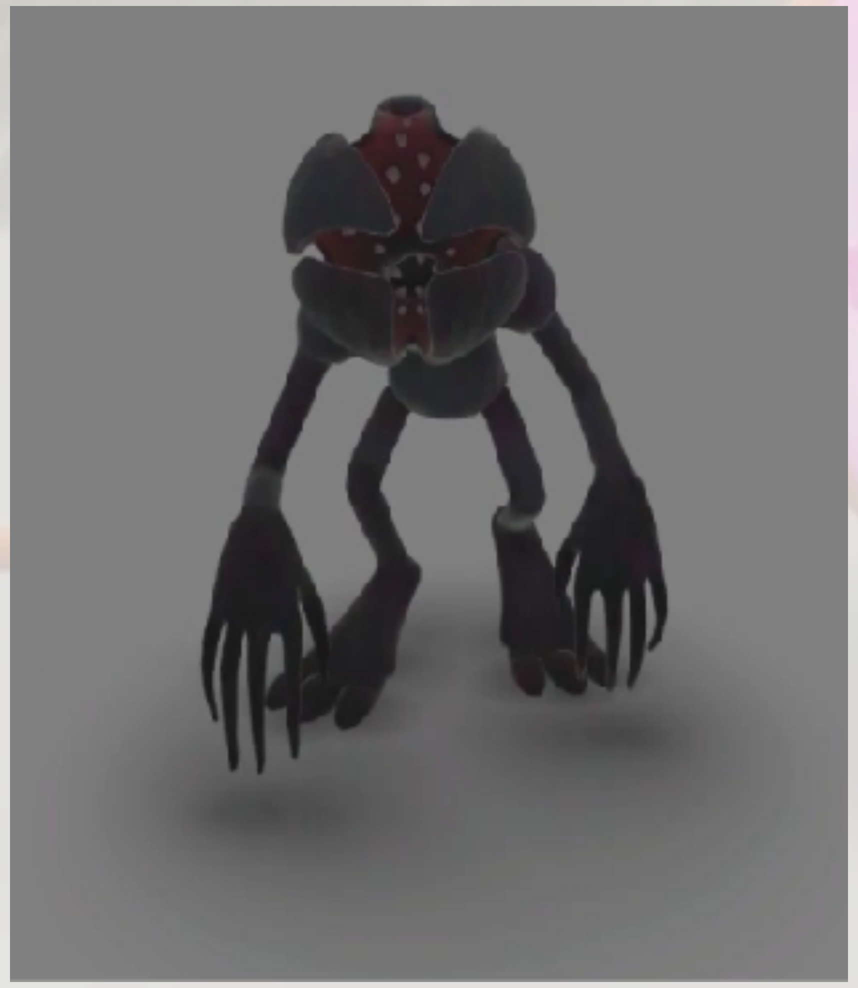


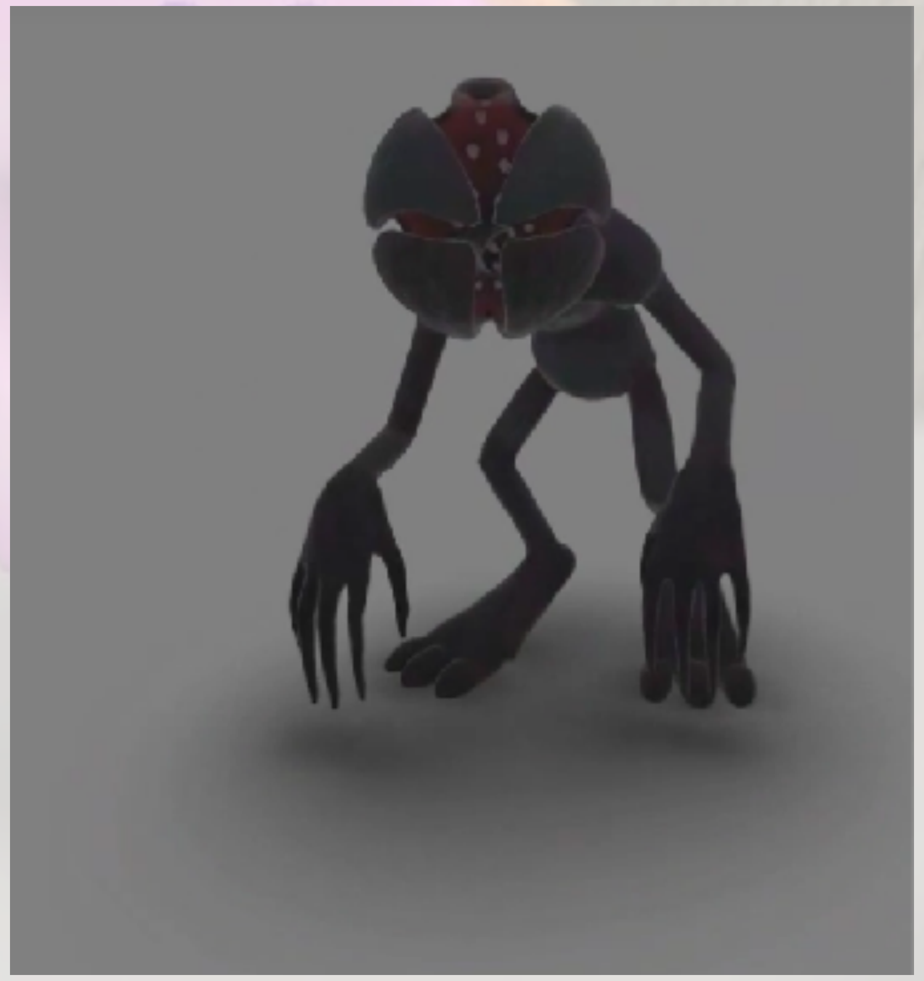
Shadows:
Blobby Shadows

Procedural shader on ground plane, driven by skeletal joints.

  - Soft base shadow (round with radial falloff).
  - More detailed contact shadows (at close proximity).
  - Combined.
- Captures relatively uniform lighting, e.g. overcast day.

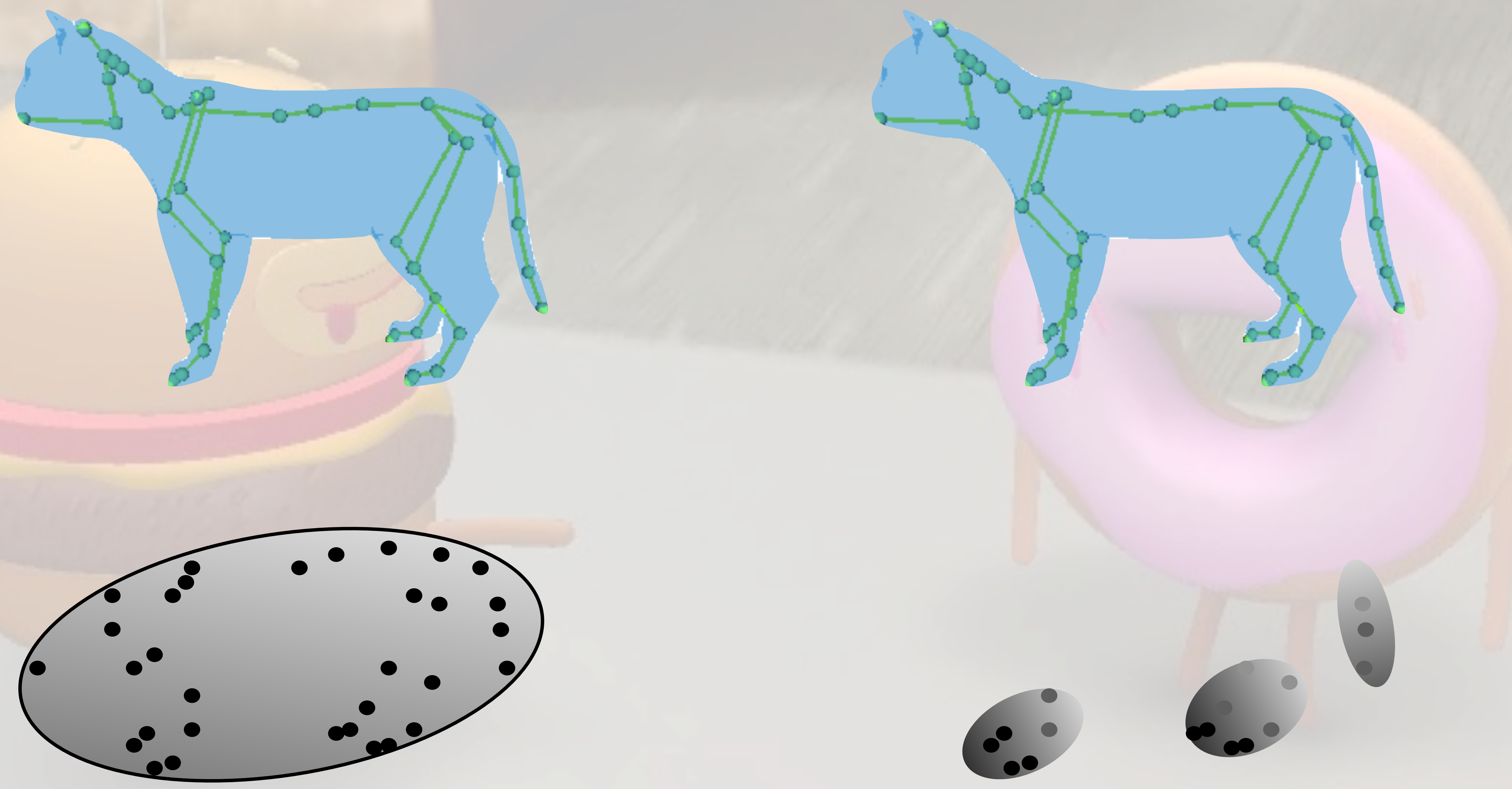




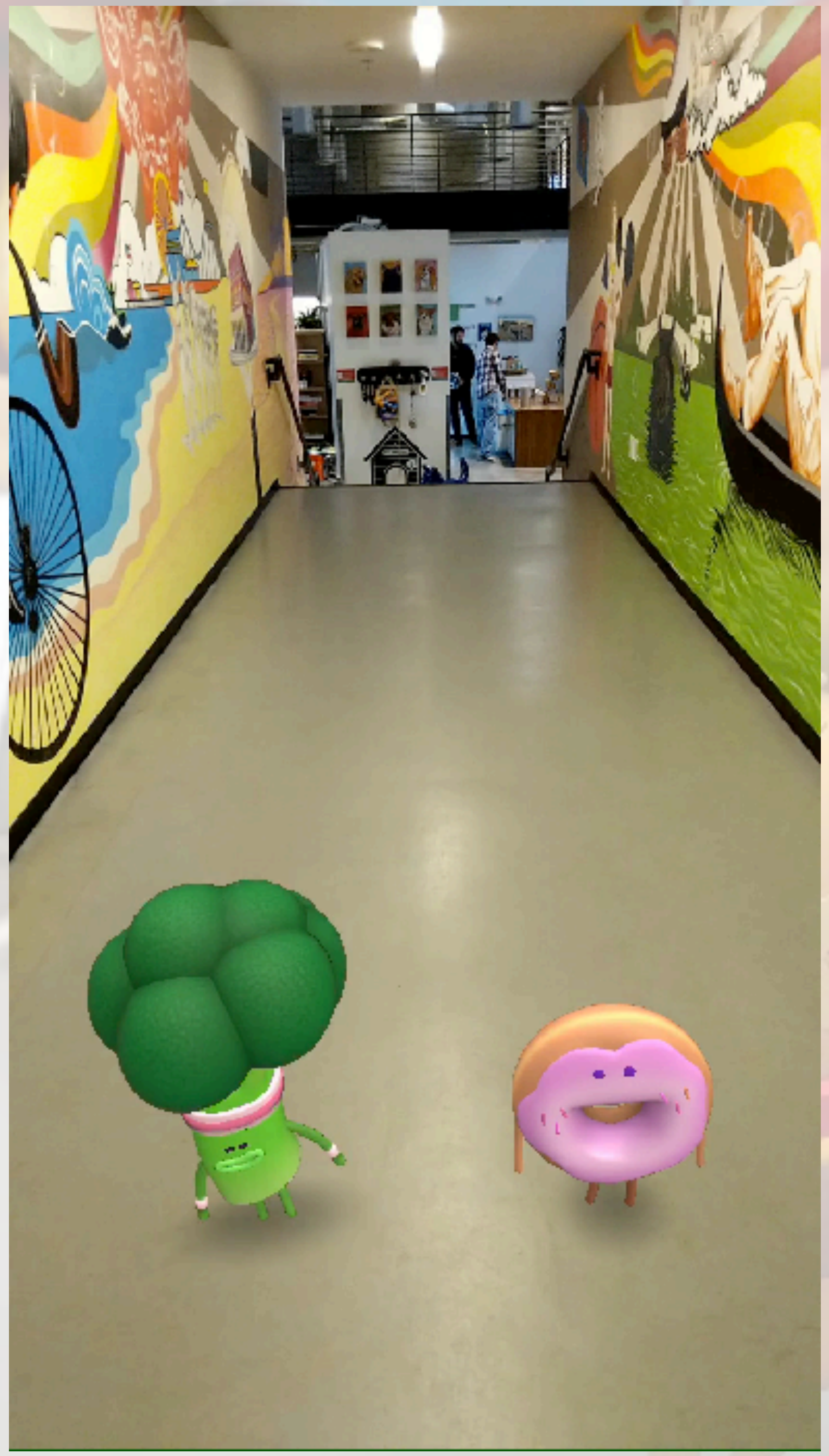


Shadows:
Blobby Shadows

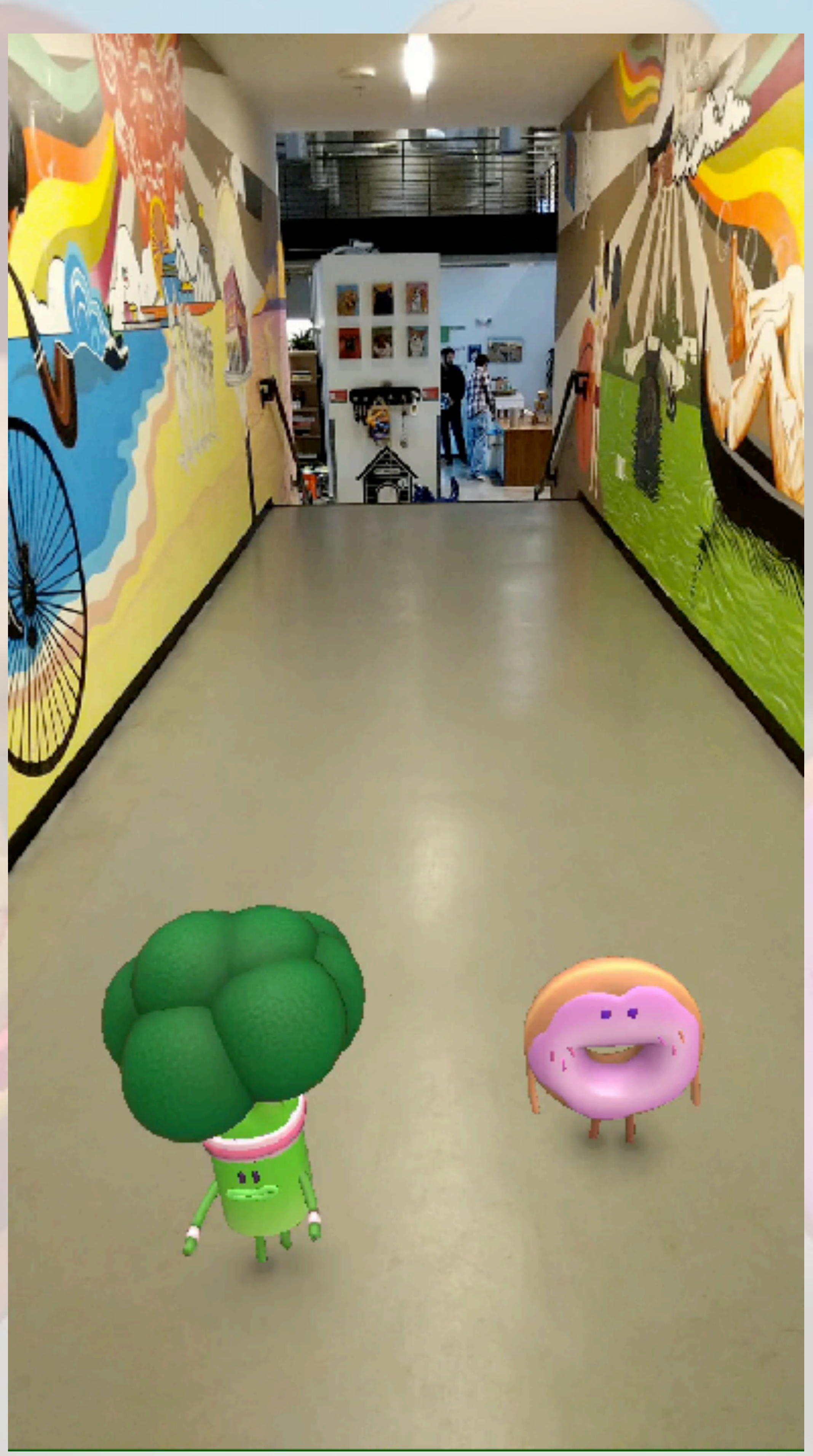
- Specific skeletal joints are designated as shadow casters.
- Base shadow: Tight-fitting ellipse containing joints.
- Contact shadow: Each joint directly darkens small region.
  - Based on distance from ground plane.



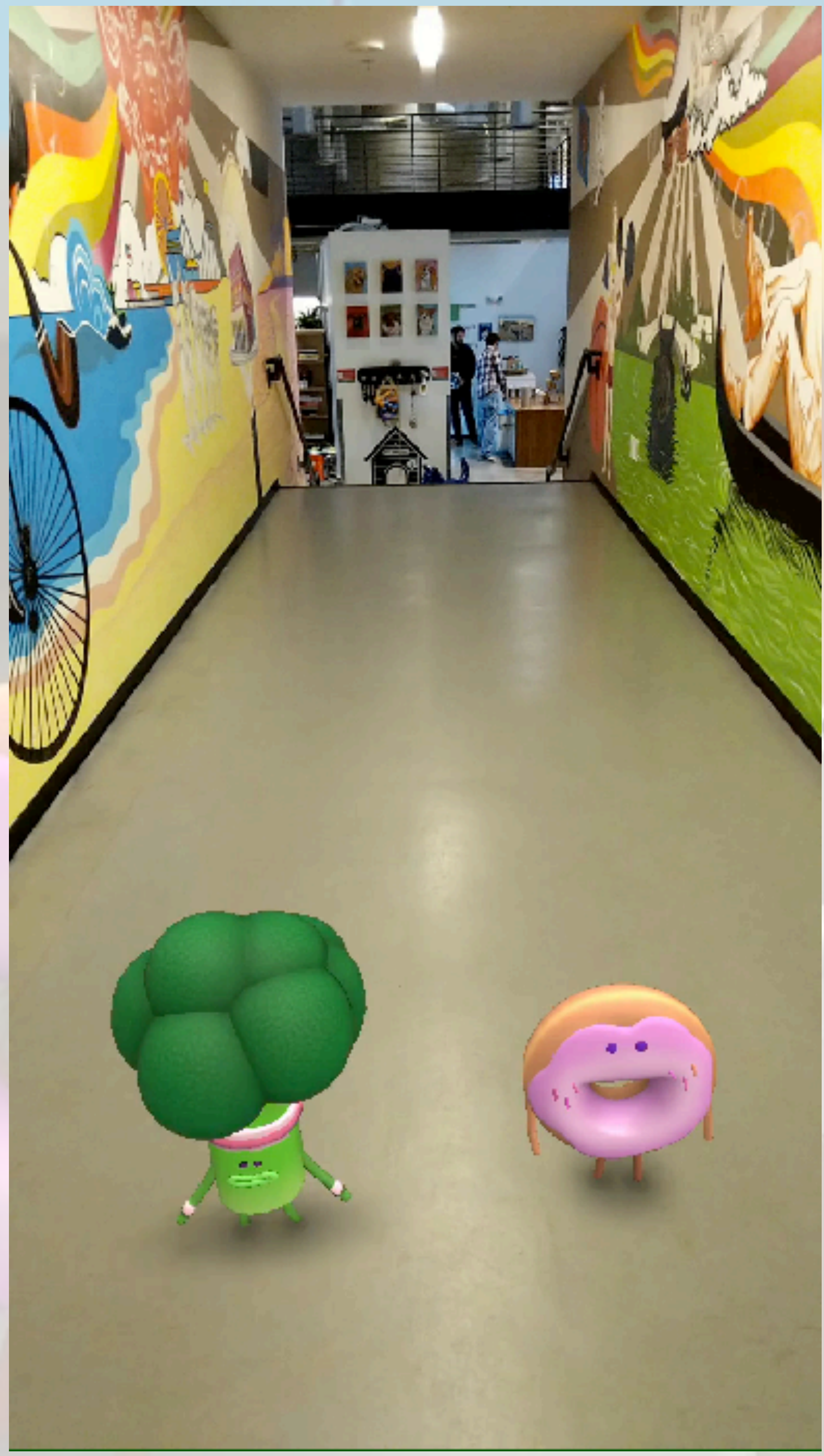
Shadows:
Blobby Base + Contact



Base



Contact

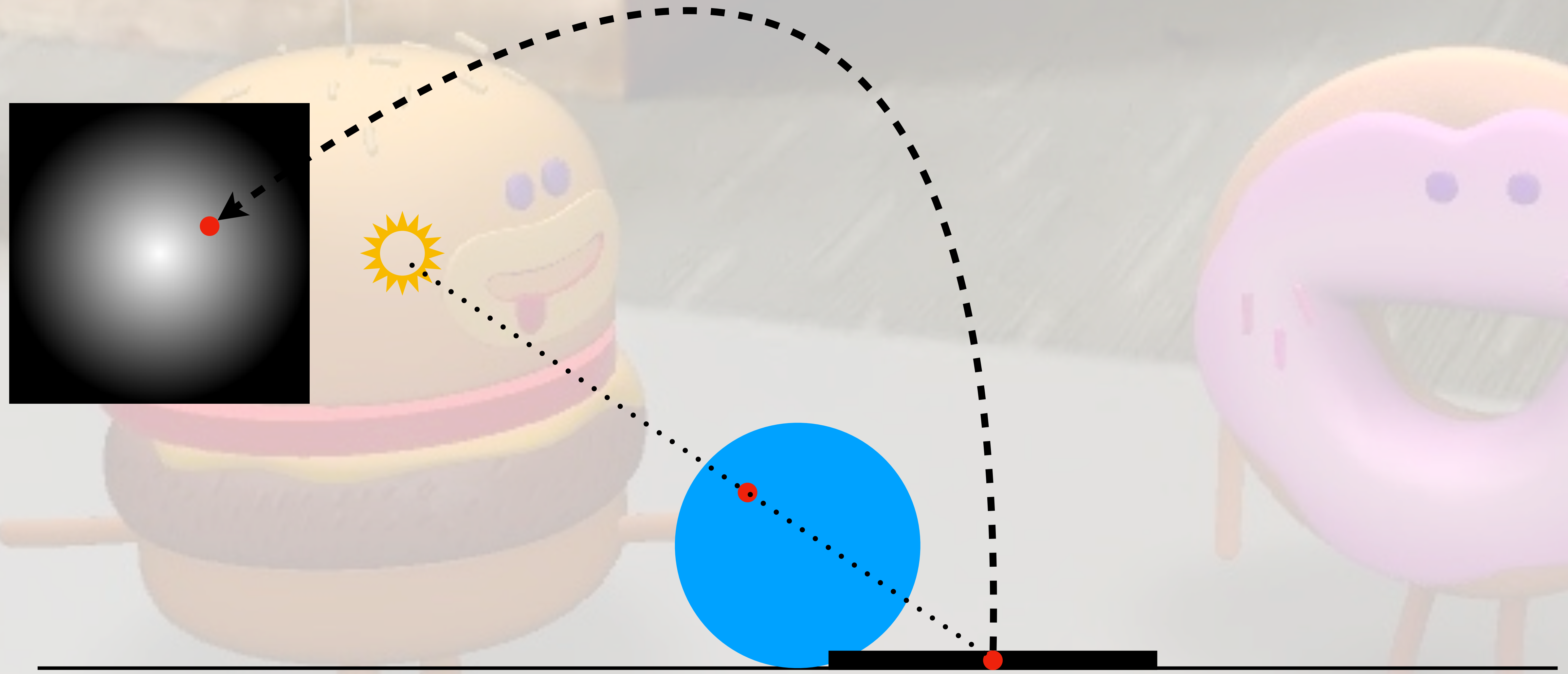


Combined

Shadows: Shadow Maps

Traditional Shadow Maps use two passes:

1. Render scene depth from light's POV into shadow map.
2. Reproject shading point into shadow map and compare stored depth to actual.



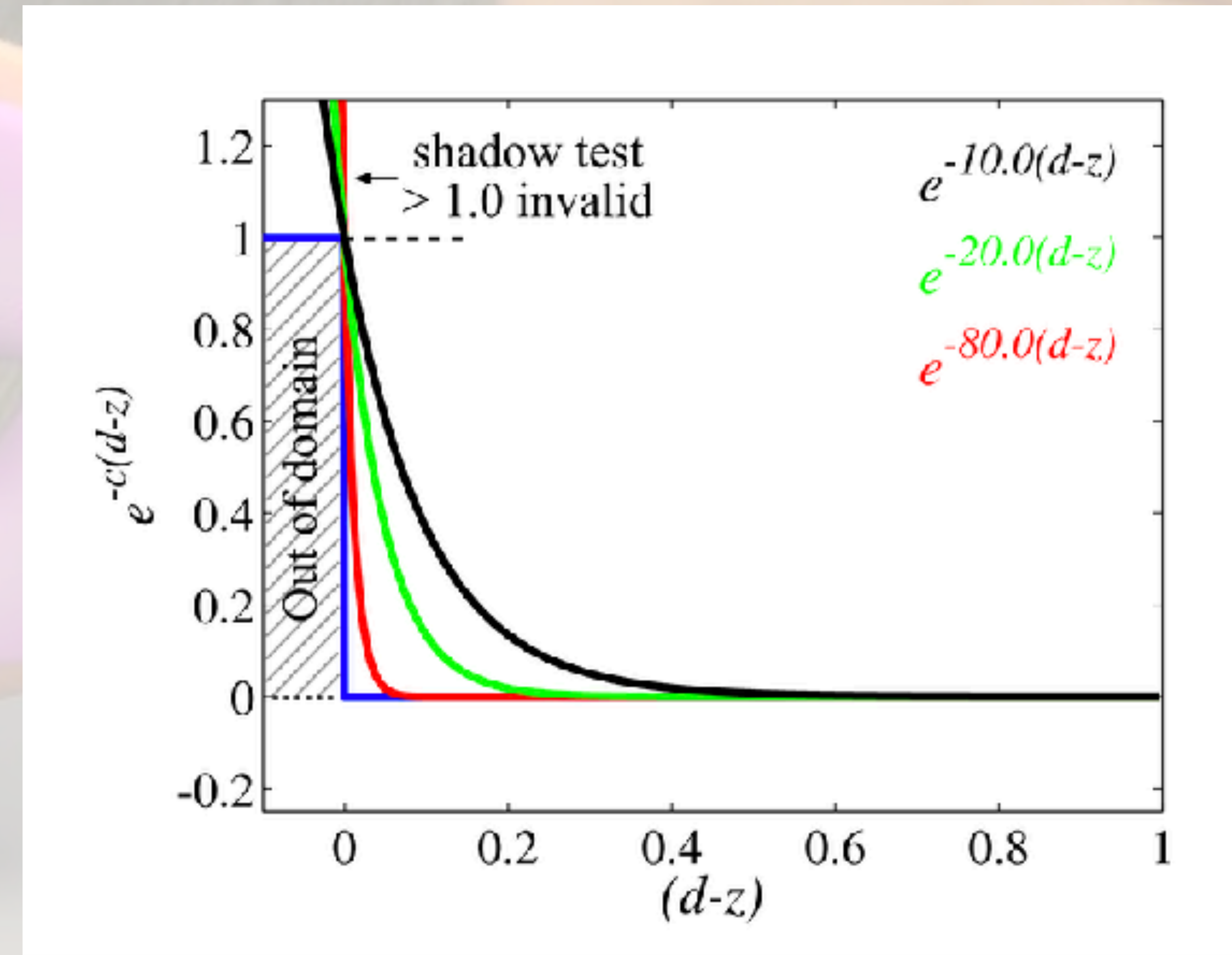
Google
Daydream

Shadows:
Exponential Shadow Maps

We implemented ***Exponential Shadow Maps*** *(ESM) [Annen et al 2008]* with an overhead light for shadow placement.

Basic idea:

- Render exponentiated depth into shadow map.
- Exponential curve approximates depth test.
  - Just a step function.
- Allows direct filtering of shadow map.



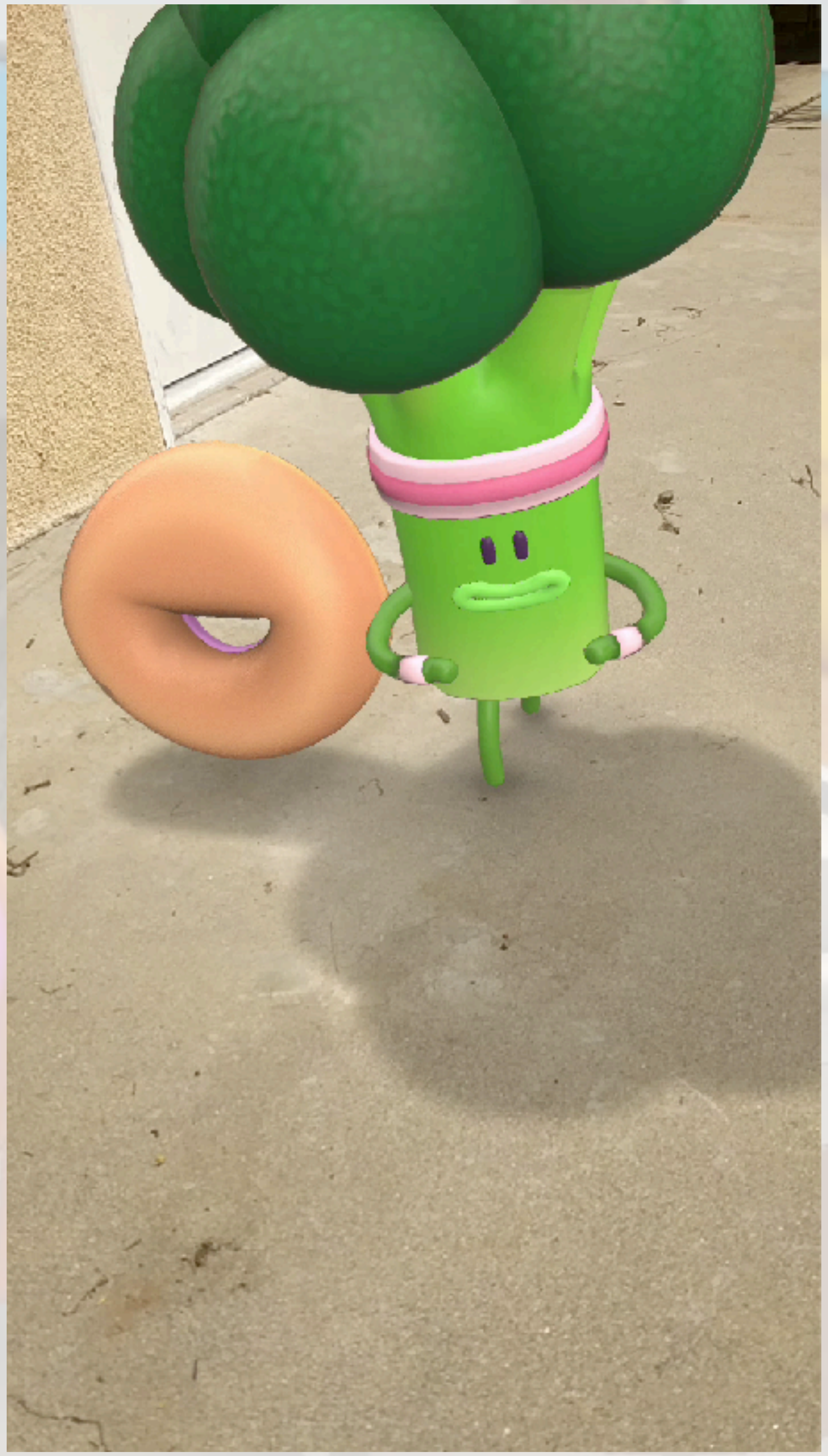
Shadows:
Exponential Shadow Maps

ESM Advantages:

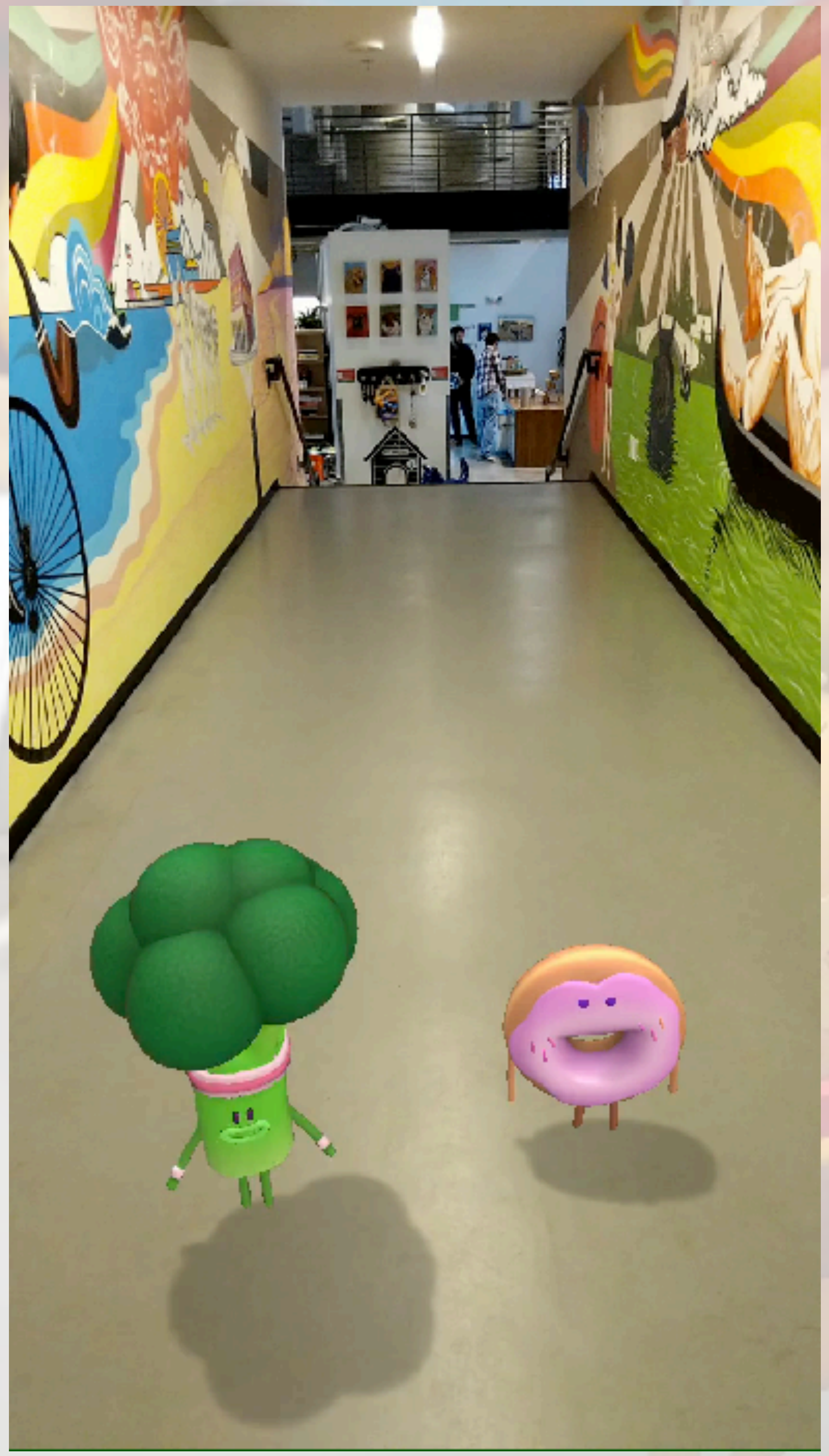
- Texture filtering of shadow map.
- Shadows can be directly blurred in an extra shader pass.
- Renderer can use mipmapping to antialias shadows.
- Less expensive / noisy than PCF.

One disadvantage:

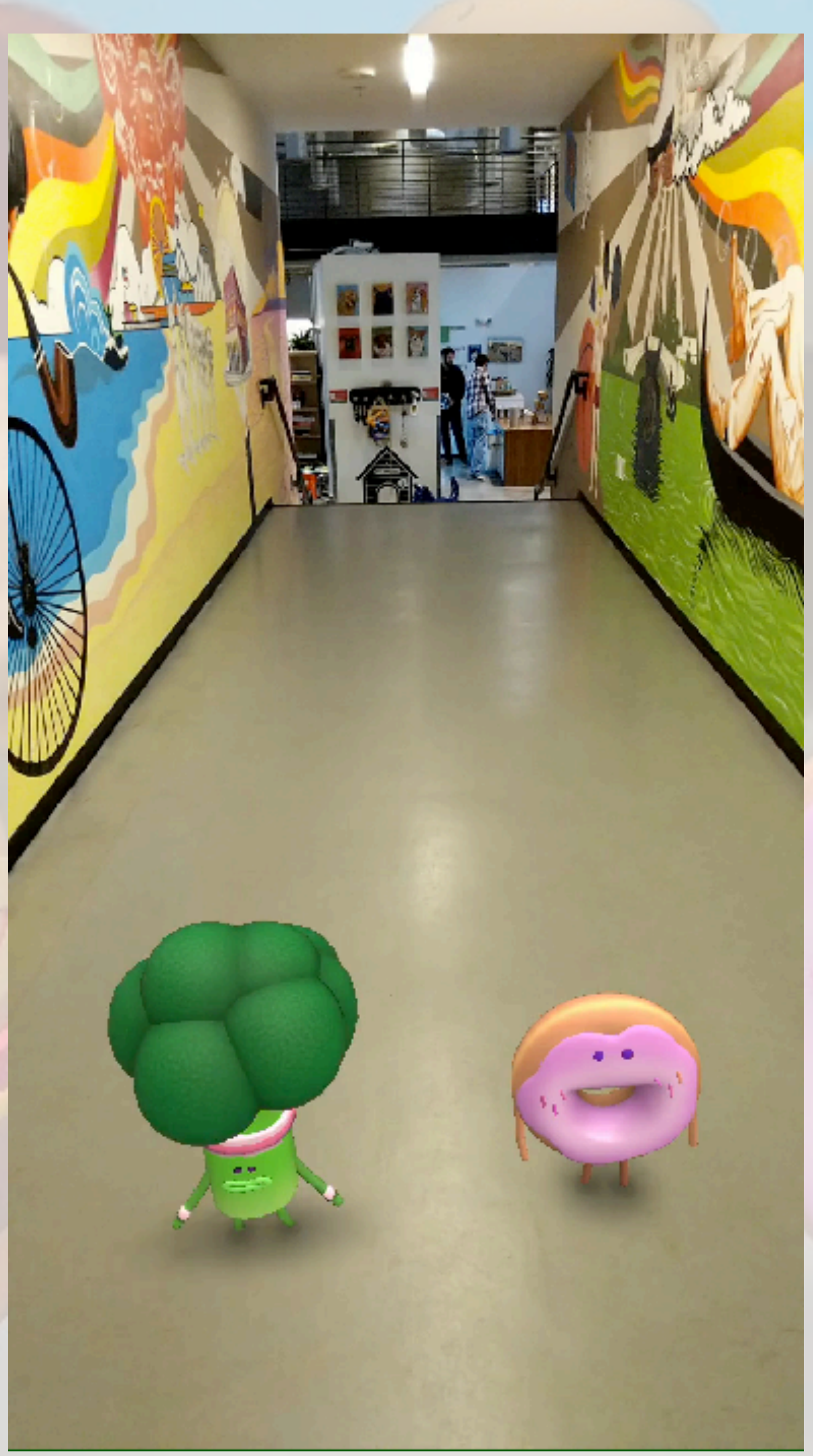
- Light leakage (but never on ground).



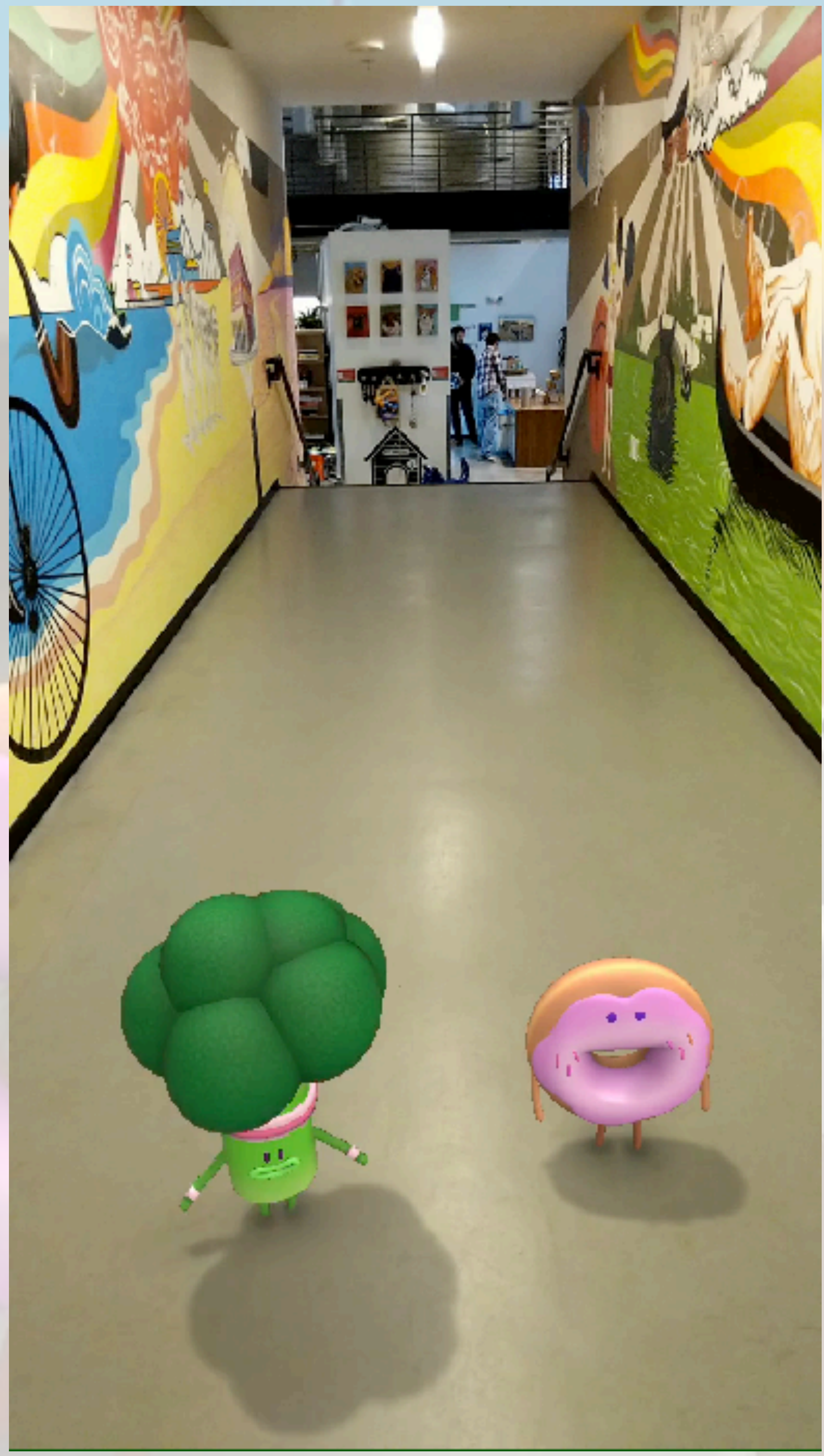
Shadows:
ESM + Blobby Shadows



ESM



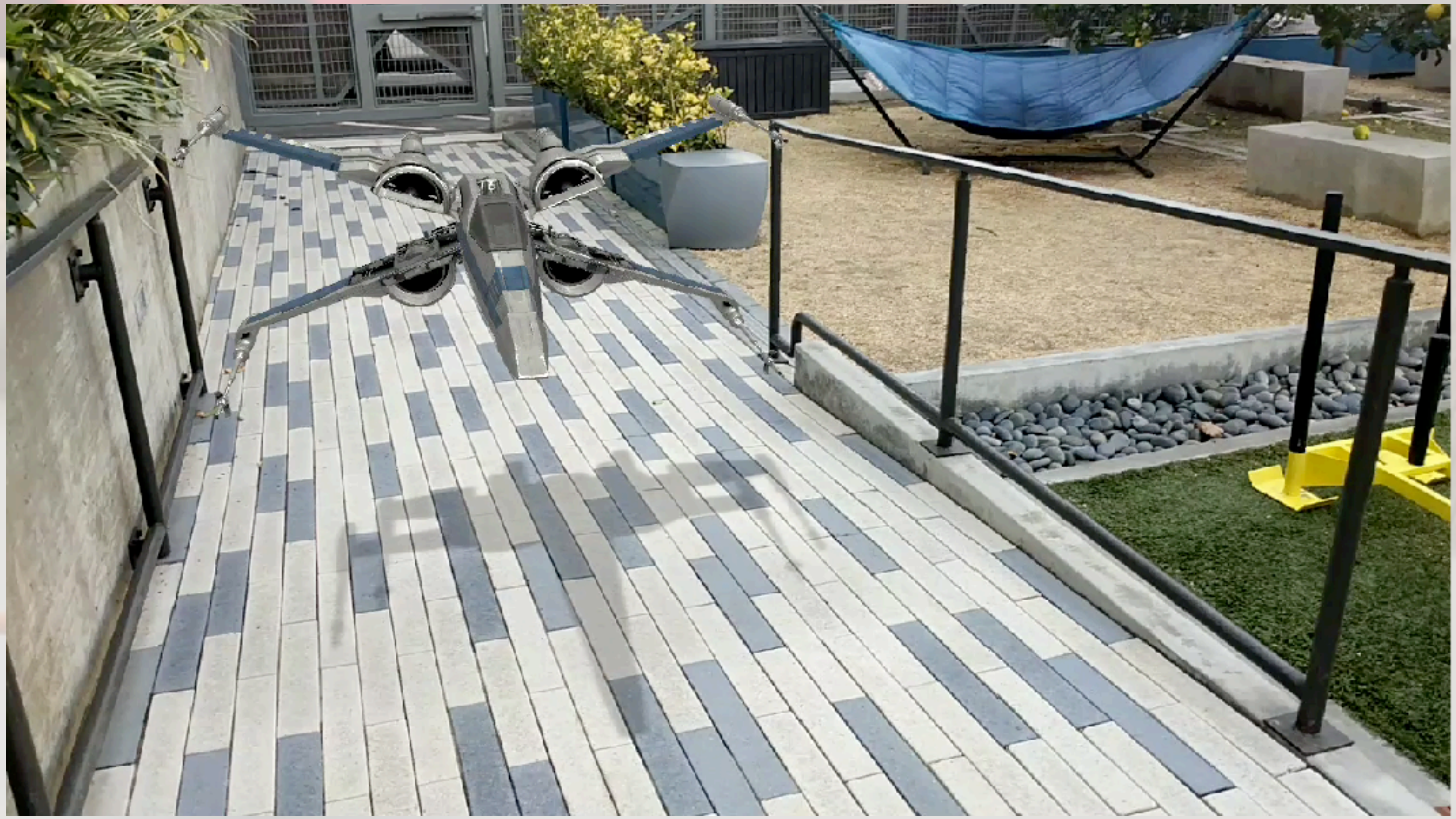
Blobby



Combined

Shadows: LOD Transition

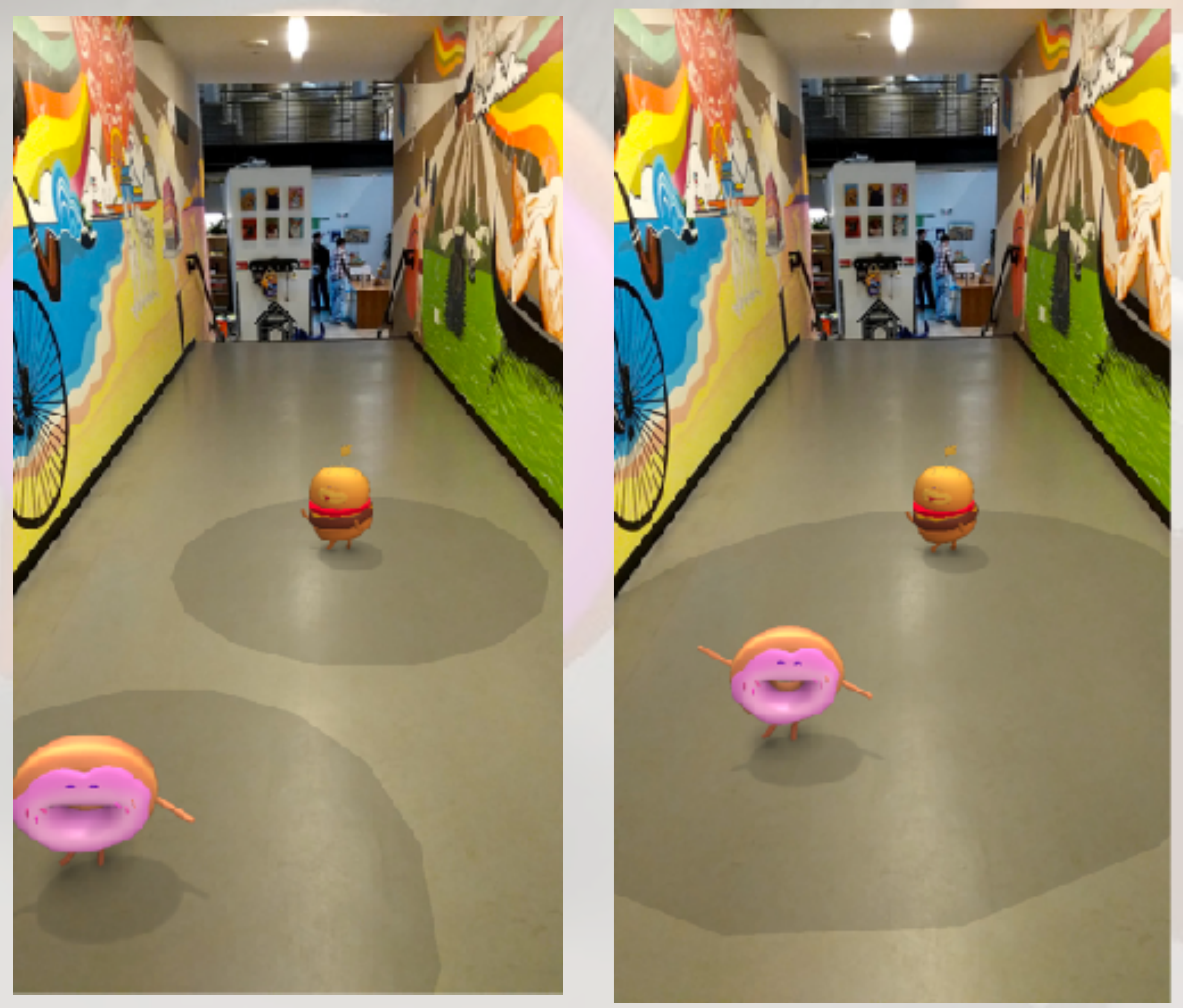
- ESM transitions to blobby base shadows far away.
- Keeps casters tightly framed in single shadow map.



Shadows: Receiver Geometry

Shadow Receiver Cards are needed for AR.

- Transparent cards, created under each sticker at AR plane height.
- Opacity varies with shadow strength.
- They follow stickers in (x, z) but not height.
- Merged when they overlap in (x,z) and are within some threshold of height (y).



Agenda

1. Development Process
2. AR Stickers Design
3. Lighting & Rendering
4. **Visual Enhancements**
5. Concluding Thoughts

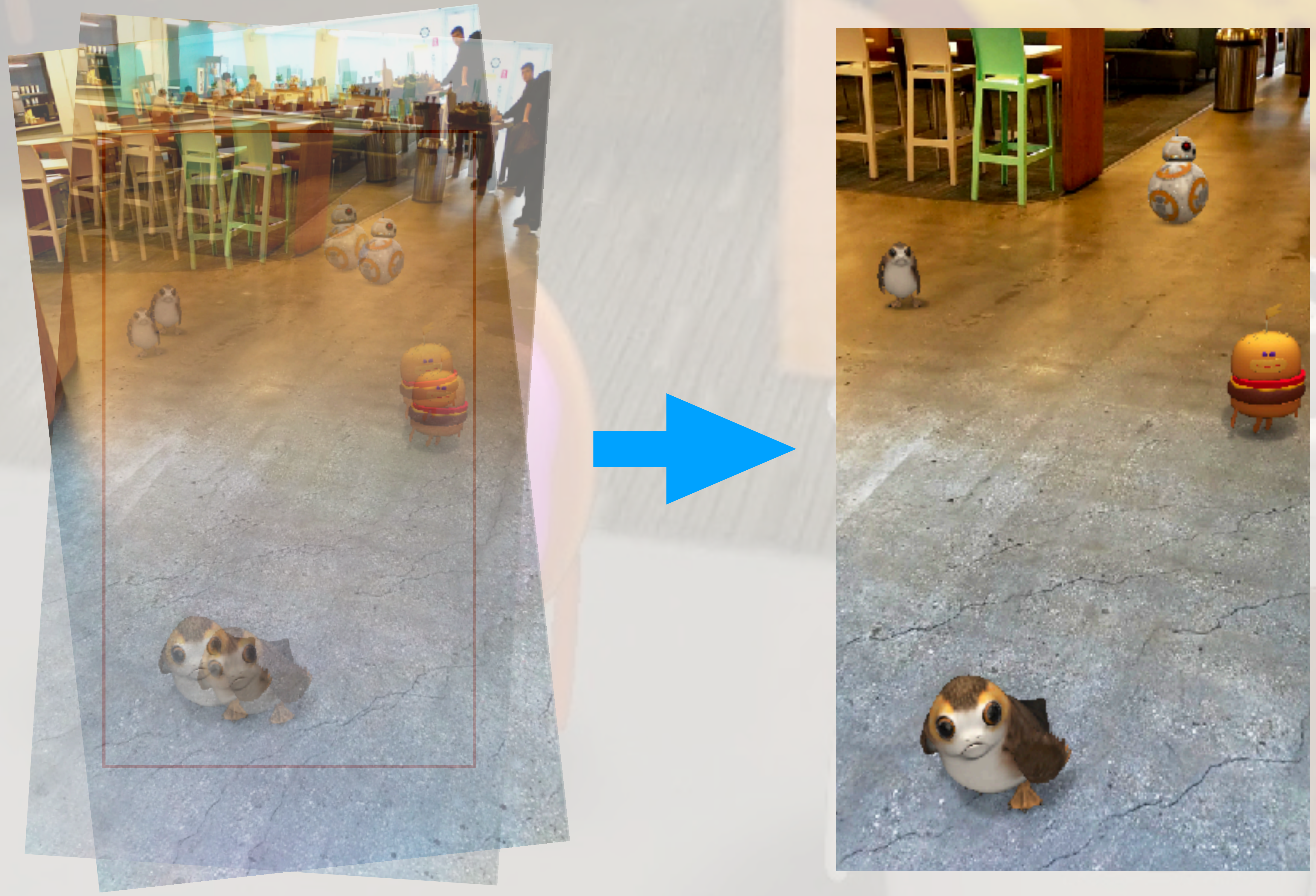
Electronic Image Stabilization

AR Stickers includes **EIS** on all Pixel phones.

- Stabilizes movement and corrects rolling shutter.
- Warps each frame, reducing effective resolution.

EIS with AR requires:

1. Stabilizing the camera feed.
2. Stabilizing the CG content (stickers, shadows, effects).



Electronic Image Stabilization

To stabilize the rendered geometry:

- Pass per-frame homography matrices into vertex shaders.
- Modify 3D position to incorporate them:
  - Project to NDC space.
  - Apply homography.
  - Unproject back to 3D (leaving depth unchanged).

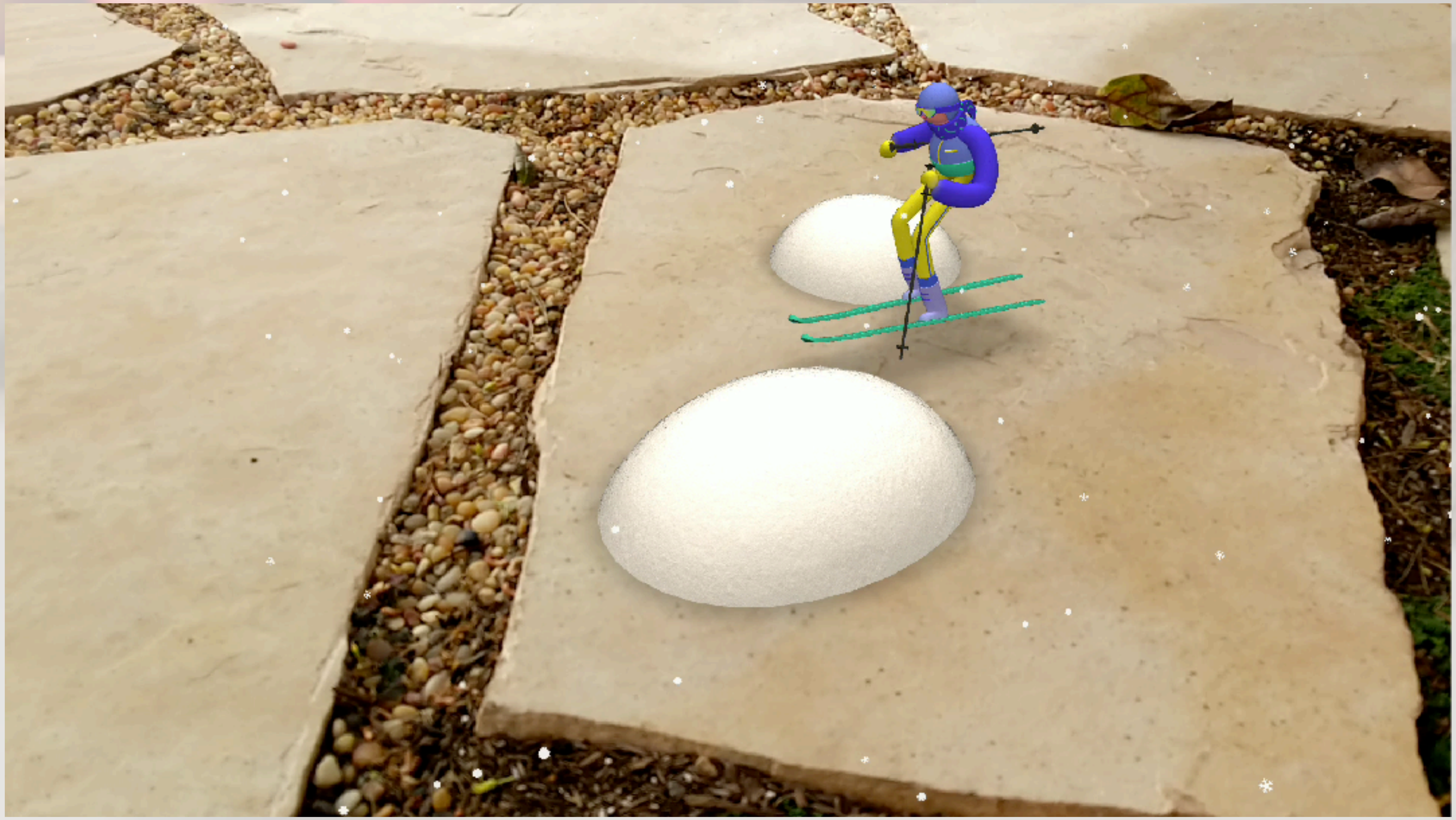
Dynamic Snow Effect

Winter Sports pack adds a falling snow effect.

- Different snowflakes chosen randomly from texture atlas.
  - Snowflake speed varies inversely with its size.
- Applied with alpha transparency onto quads.

Snow Effect

- Wind effect matches skier motion.
- Simulated moguls part of model animation.



Agenda

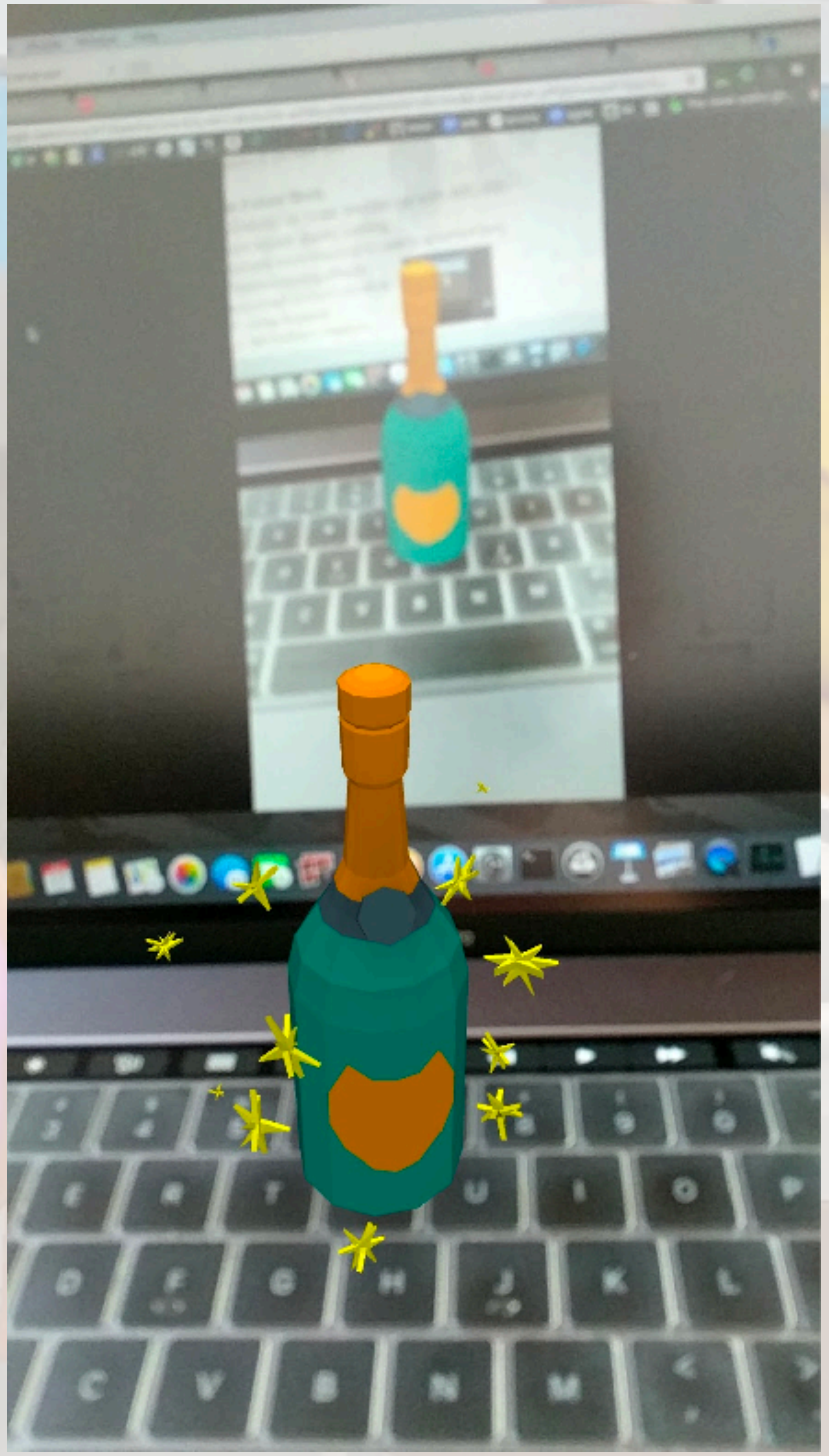
1. Development Process
2. AR Stickers Design
3. Lighting & Rendering
4. Visual Enhancements
5. **Concluding Thoughts**

Concluding Thoughts

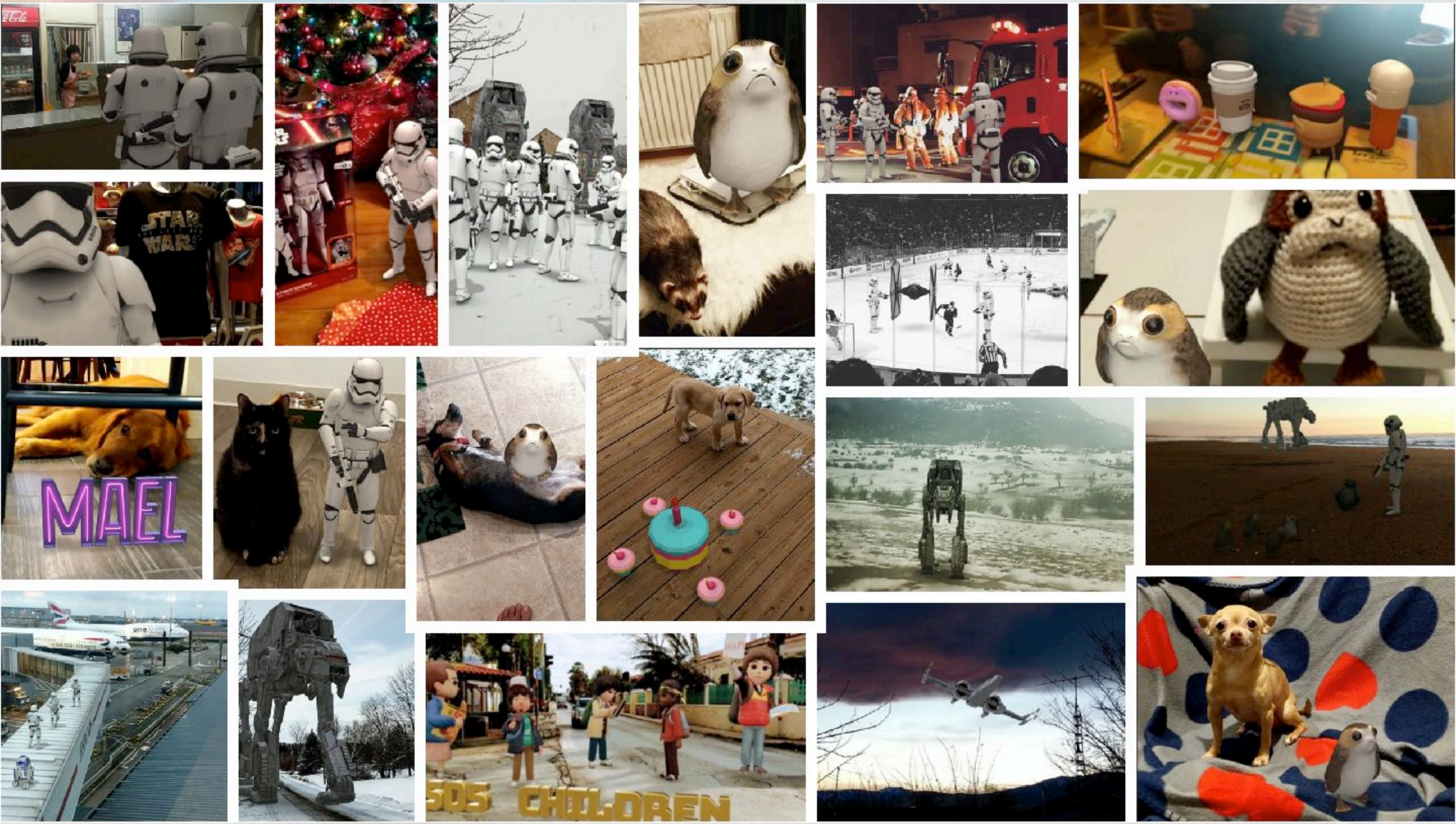
AR Stickers has been well received!

- androidcentral, cnet, engadget, techcrunch, the verge
- Millions of stickers placed by our users.
- Millions of photos & videos captured.

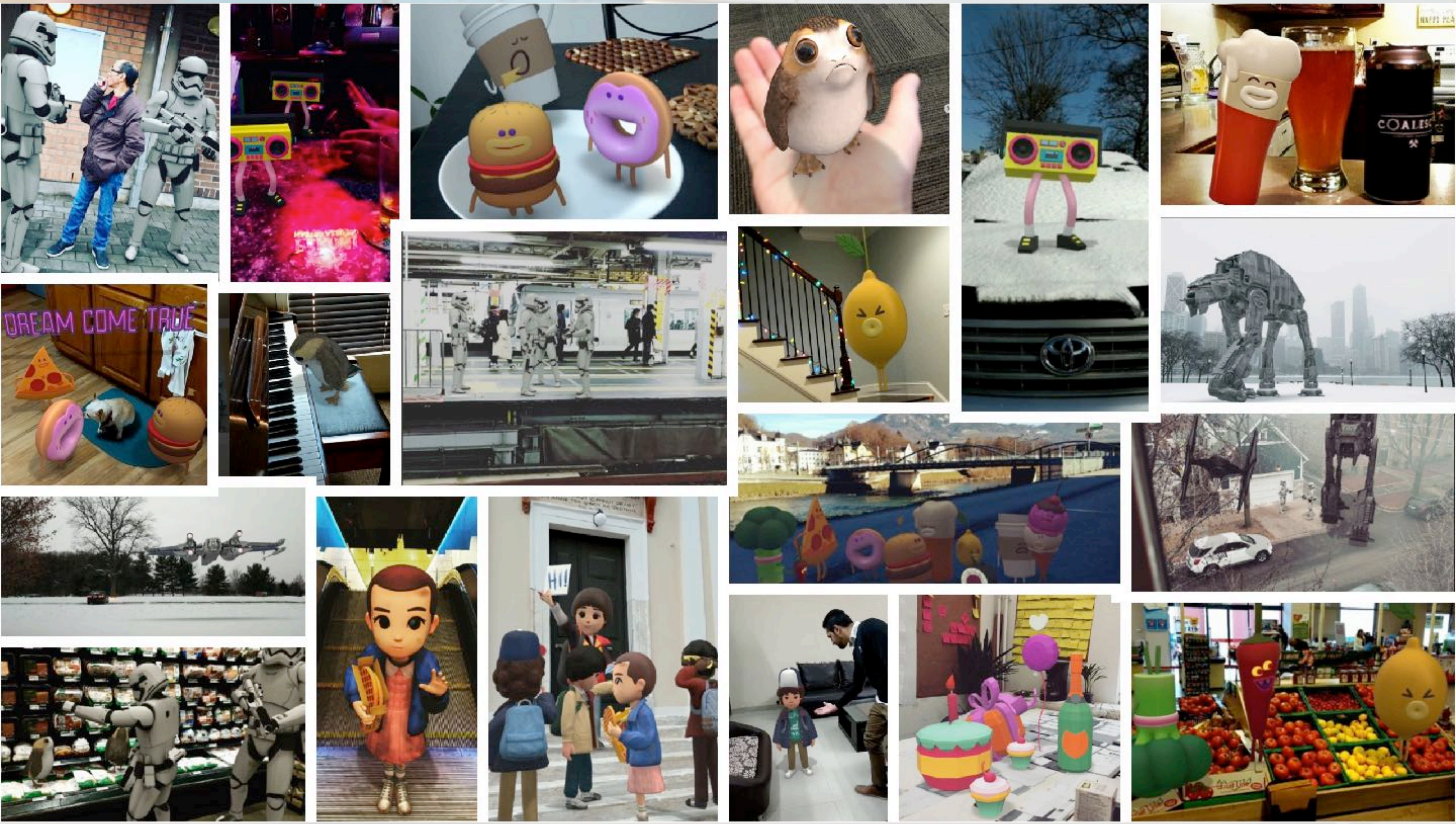
- **Play Store ratings**
  - AR Stickers: 4.4 stars
  - Blocks Pack: 4.9 stars
  - Foodmoji Pack: 5.0 stars
  - Text Pack: 5.0 stars
  - Winter Sports Pack: 4.5 stars



Concluding Thoughts



Concluding Thoughts



Concluding Thoughts

Google

Daydream

THANK YOU