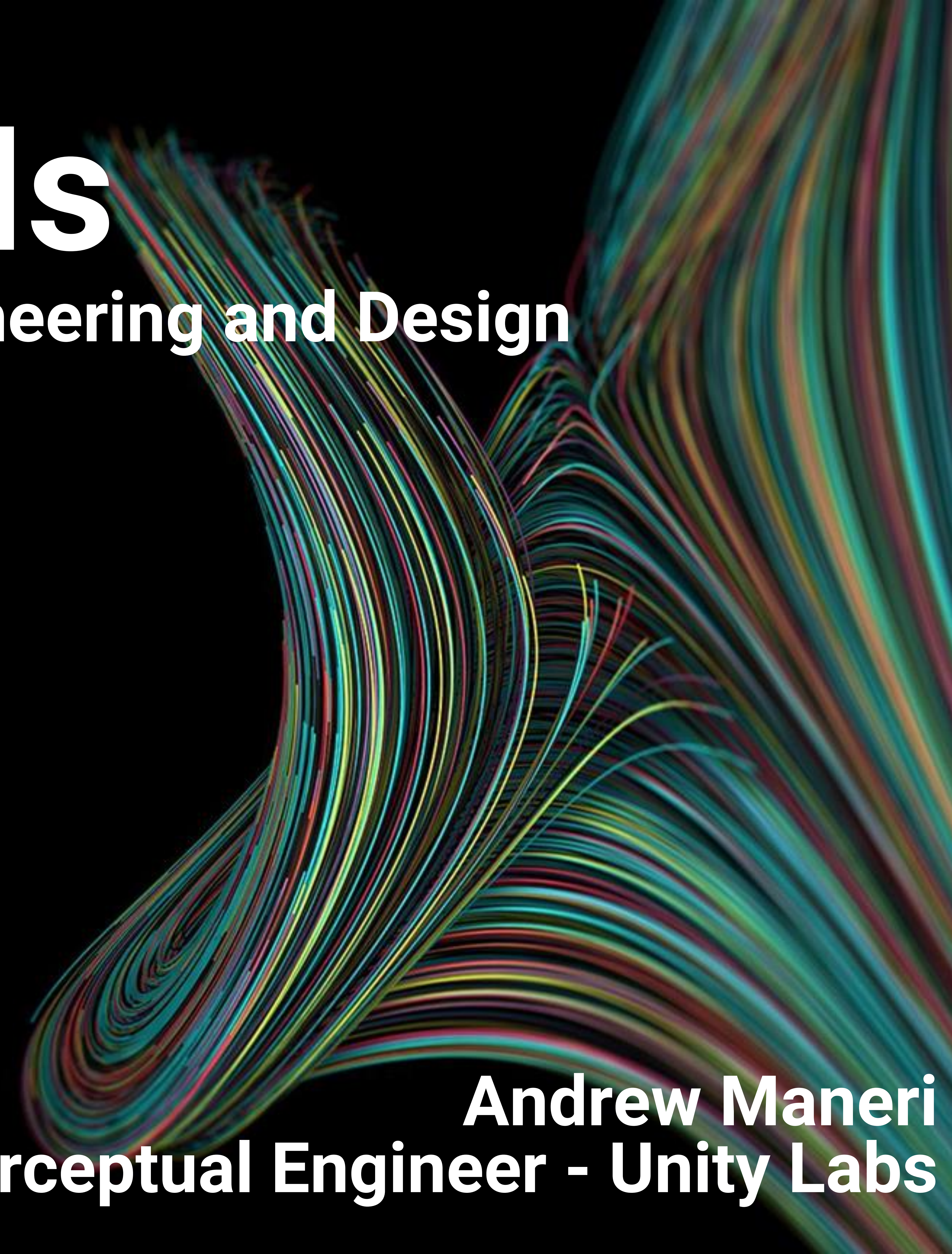
Reasoning APIs

How to Translate AR Between Engineering and Design



Generative Art – Made with Unity

Andrew Maneri
Perceptual Engineer - Unity Labs

Unity Labs

Artificial intelligence

Design

Machine learning

Spatial computing

Animation

Behaviors

Creation tools

Unity in the future

Unity Labs

Artificial intelligence

Design

Machine learning

Spatial computing

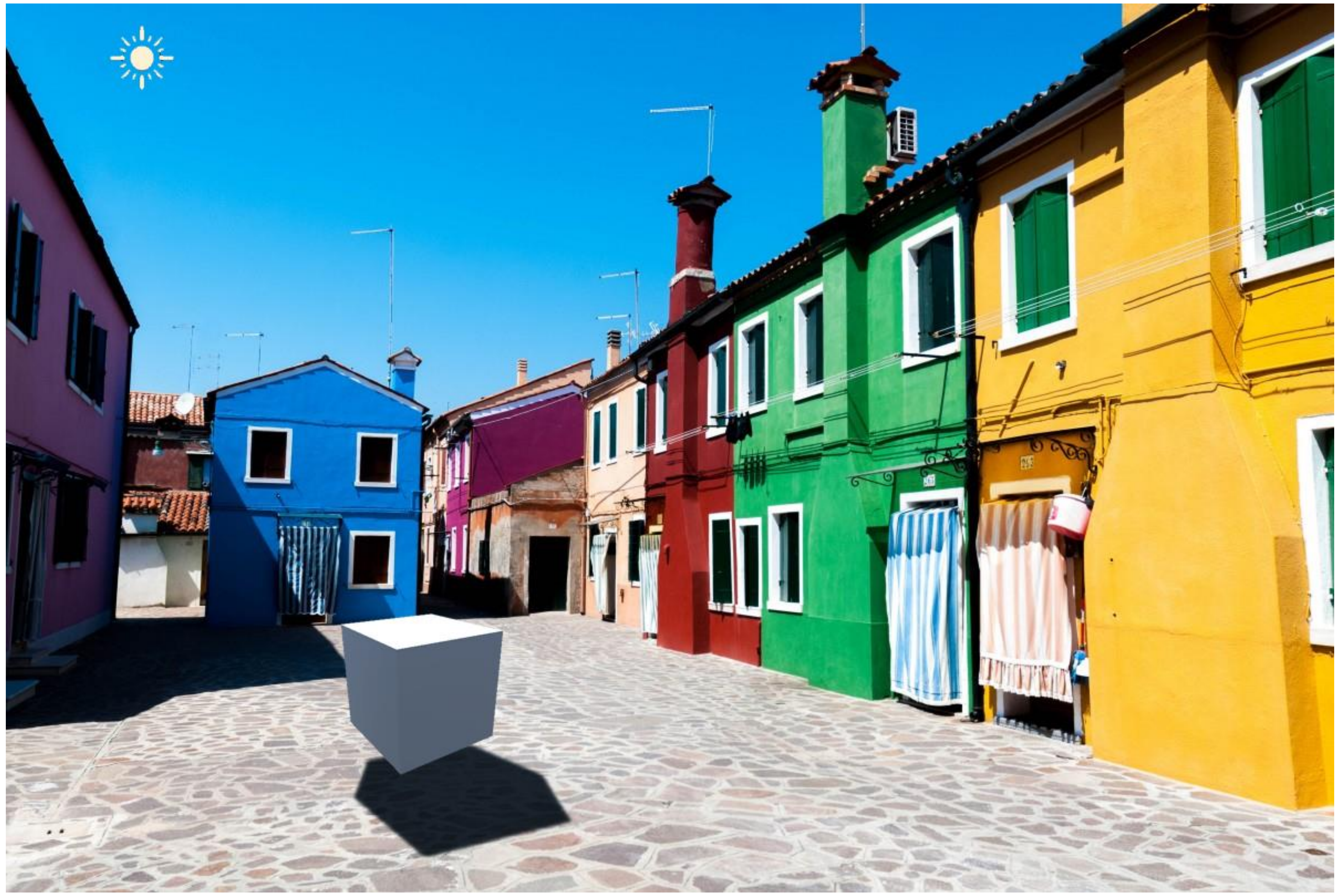
Animation

Behaviors

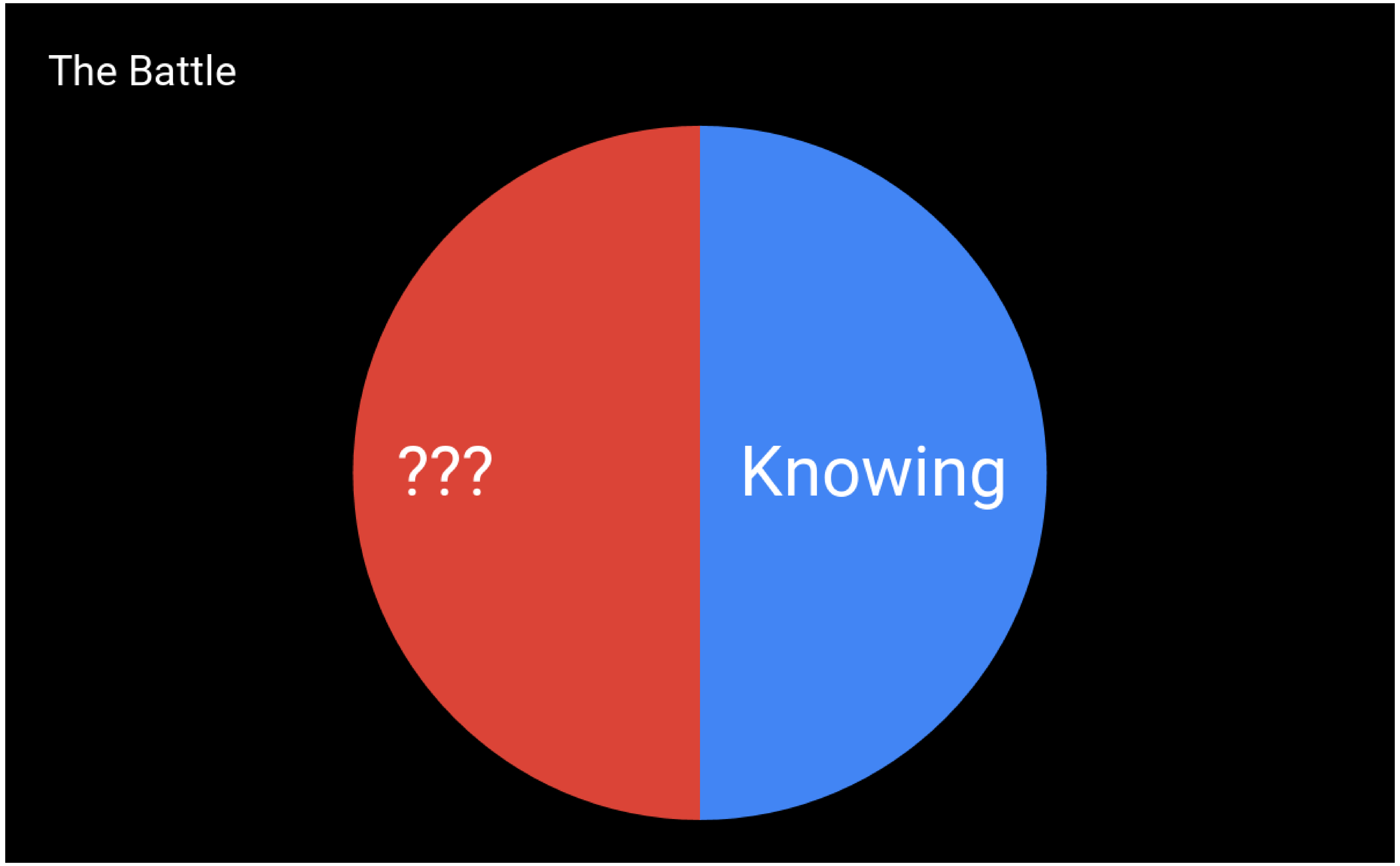
Creation tools

Unity in the future

Example: AR Re-lighting

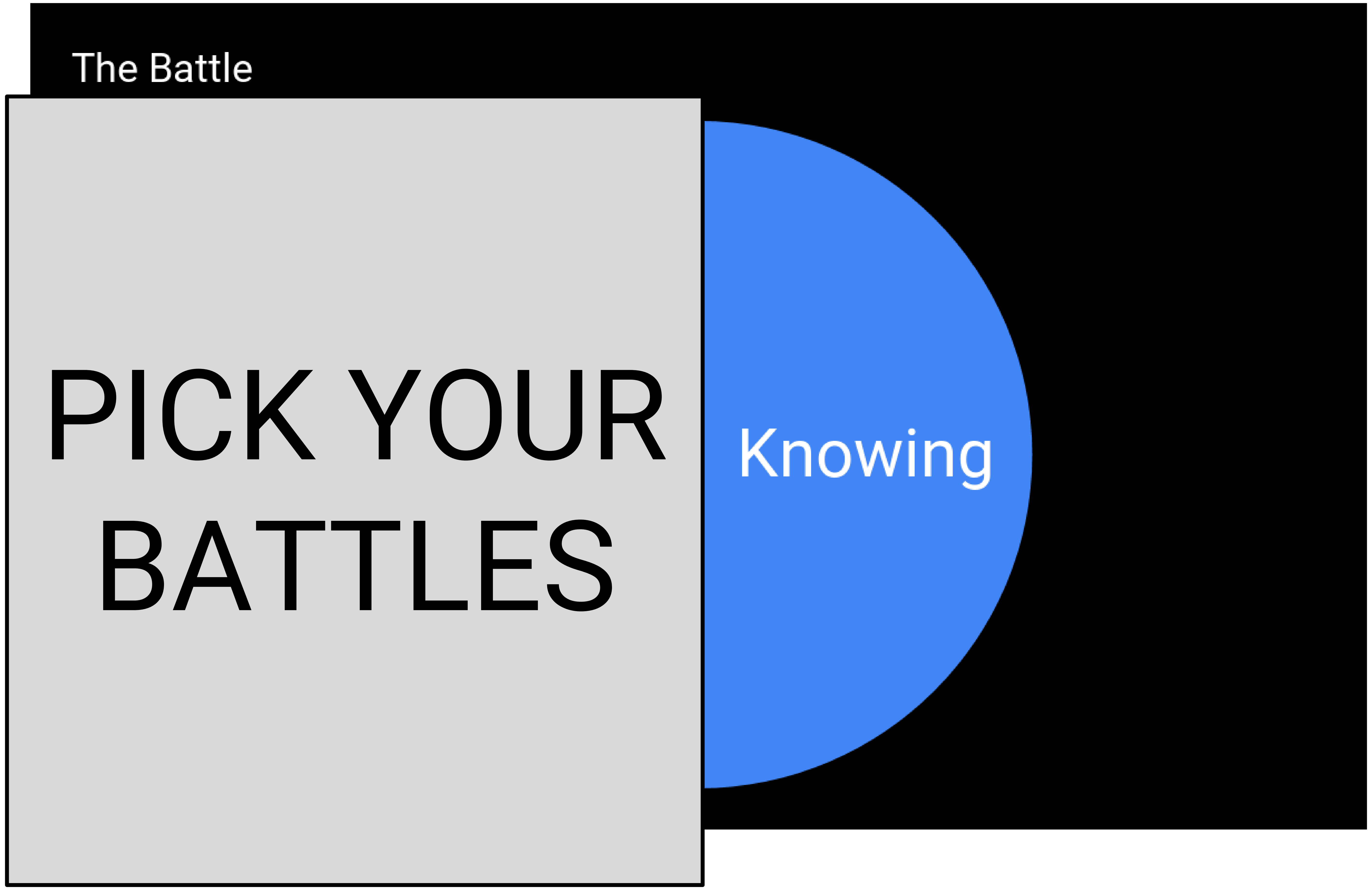


The Other Half of the Battle



The Battle

The Other Half of the Battle

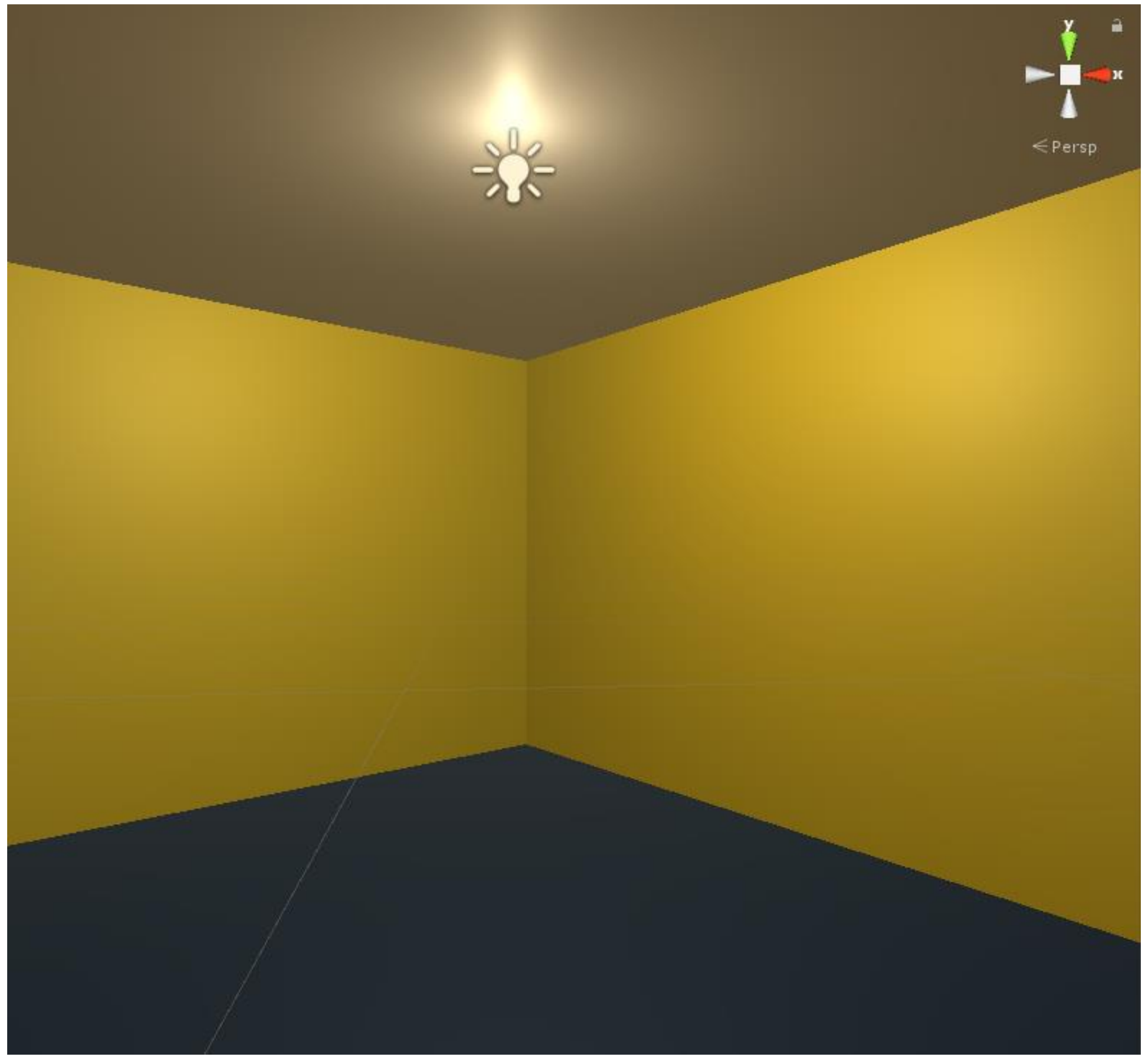


The Battle

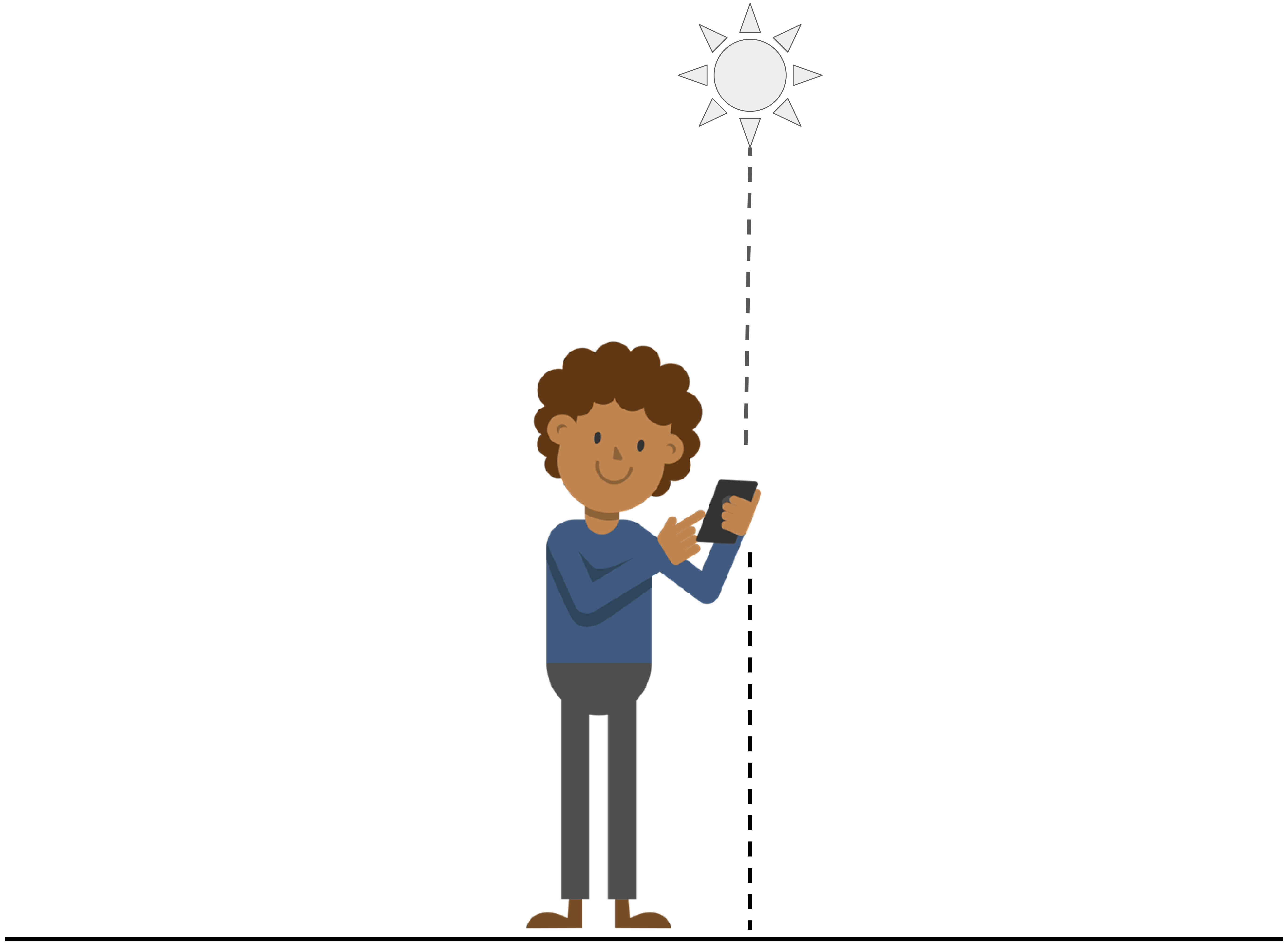
Knowing

PICK YOUR BATTLES

AR Re-lighting - Made Easy



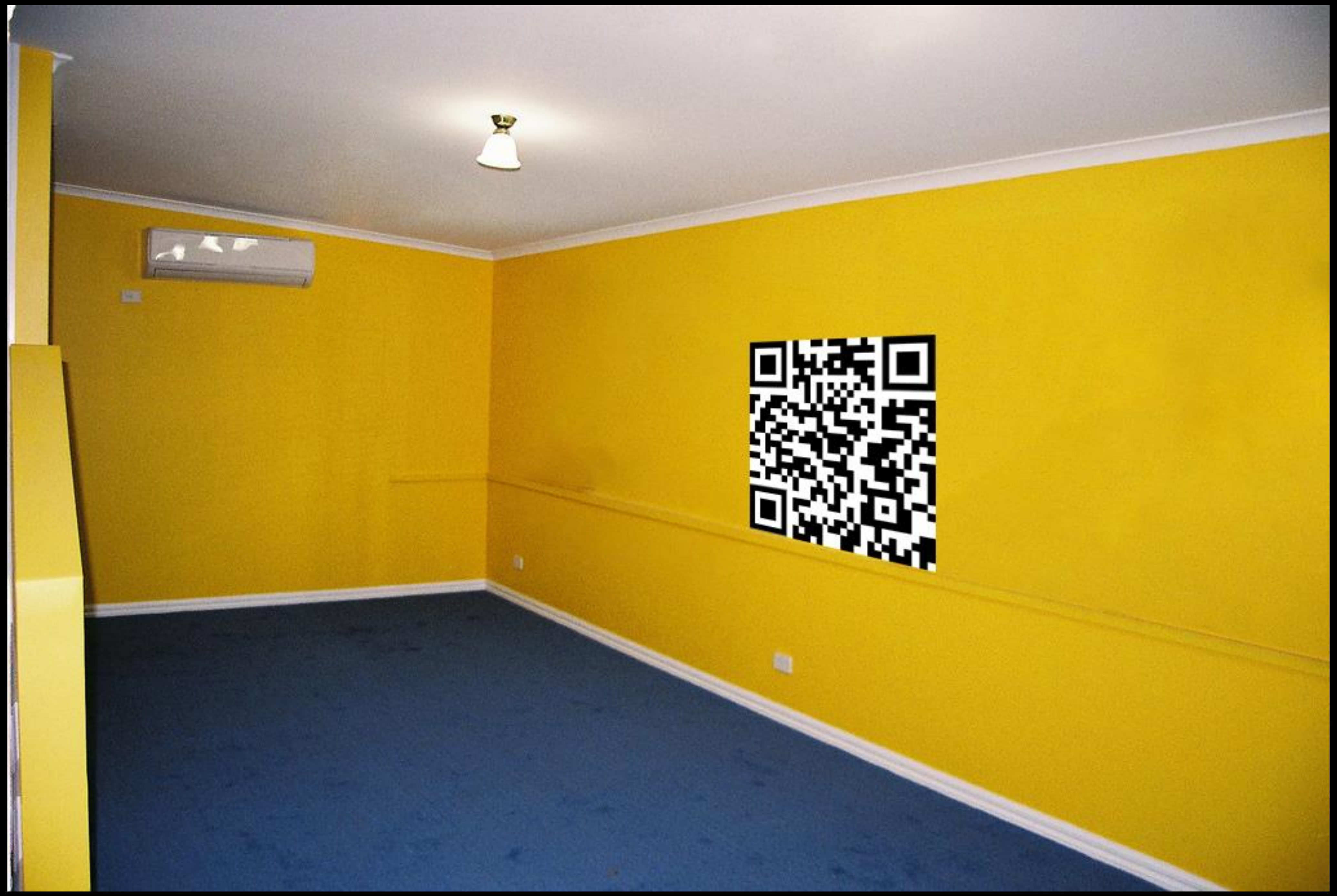
AR Re-lighting - Made Easy



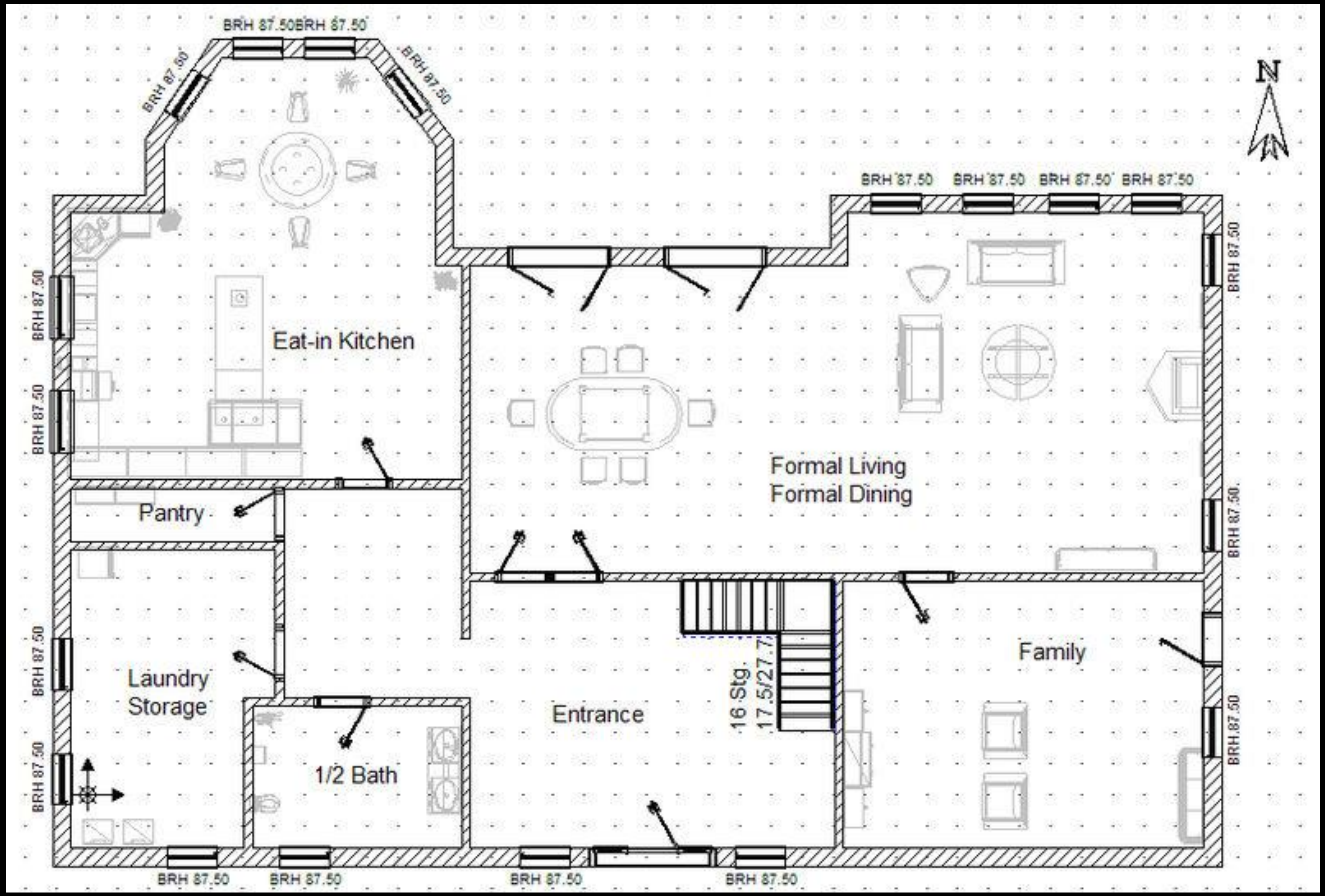
The *Real* Challenge of AR

View example video at: https://youtu.be/tUsVKQae7A8

...Seriously, this is AR



More steps forward



Even more steps forward

Select blueprints from database

Use time to estimate sun direction

Use the phone video

The Future of AR

Small, bespoke solutions

Context needed

Selected in real time

The *Real* Challenge of AR

Design: Intuitive authoring without knowing…

- What data we have
- Where it comes from
- Its format

The *Real* Challenge of AR

Engineering: How to determine...

- Which solution(s) to use
- When to scale up or down
- How to keep implementation internal

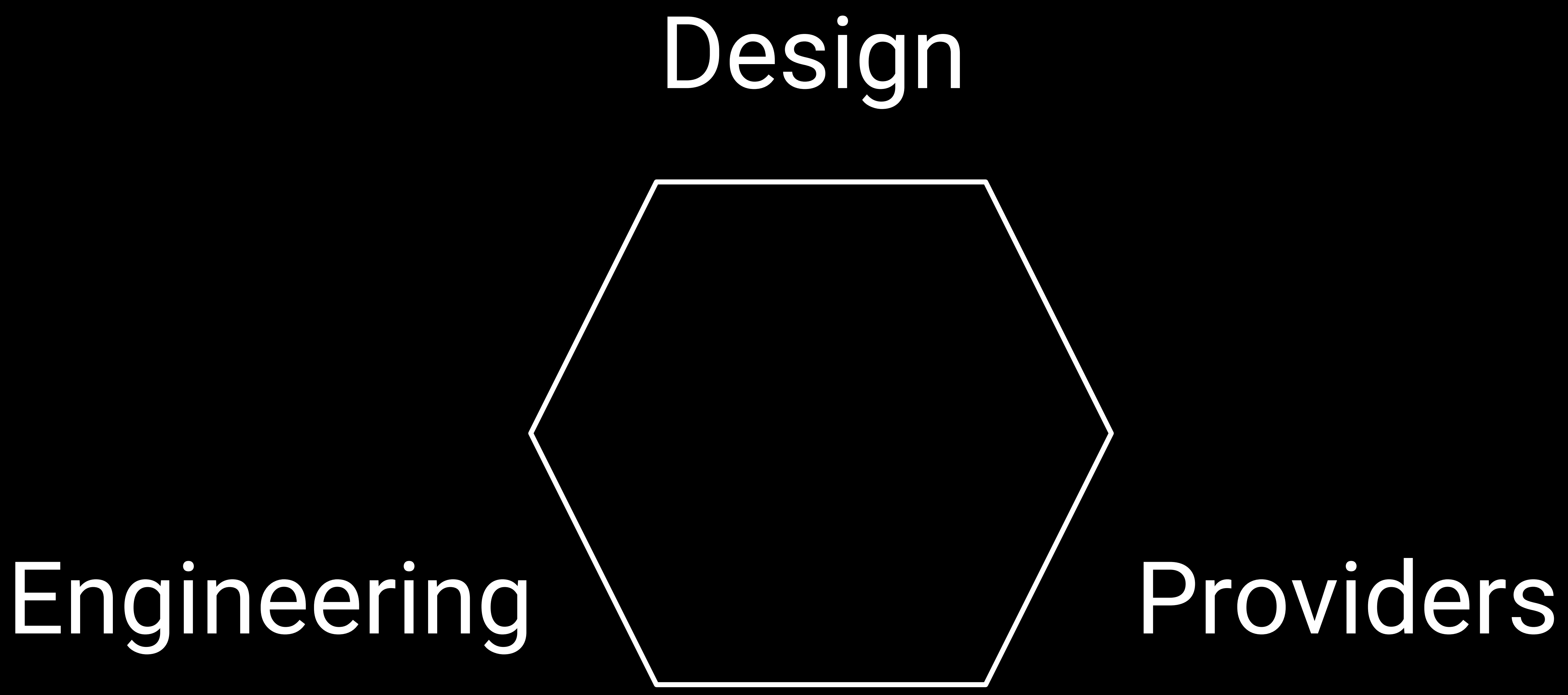
unity

The *Real* Challenge of AR

Providers: How to provide data that is...

- Simple for designers
- Robust for engineers
- Is ‘future proof’

Reasoning API

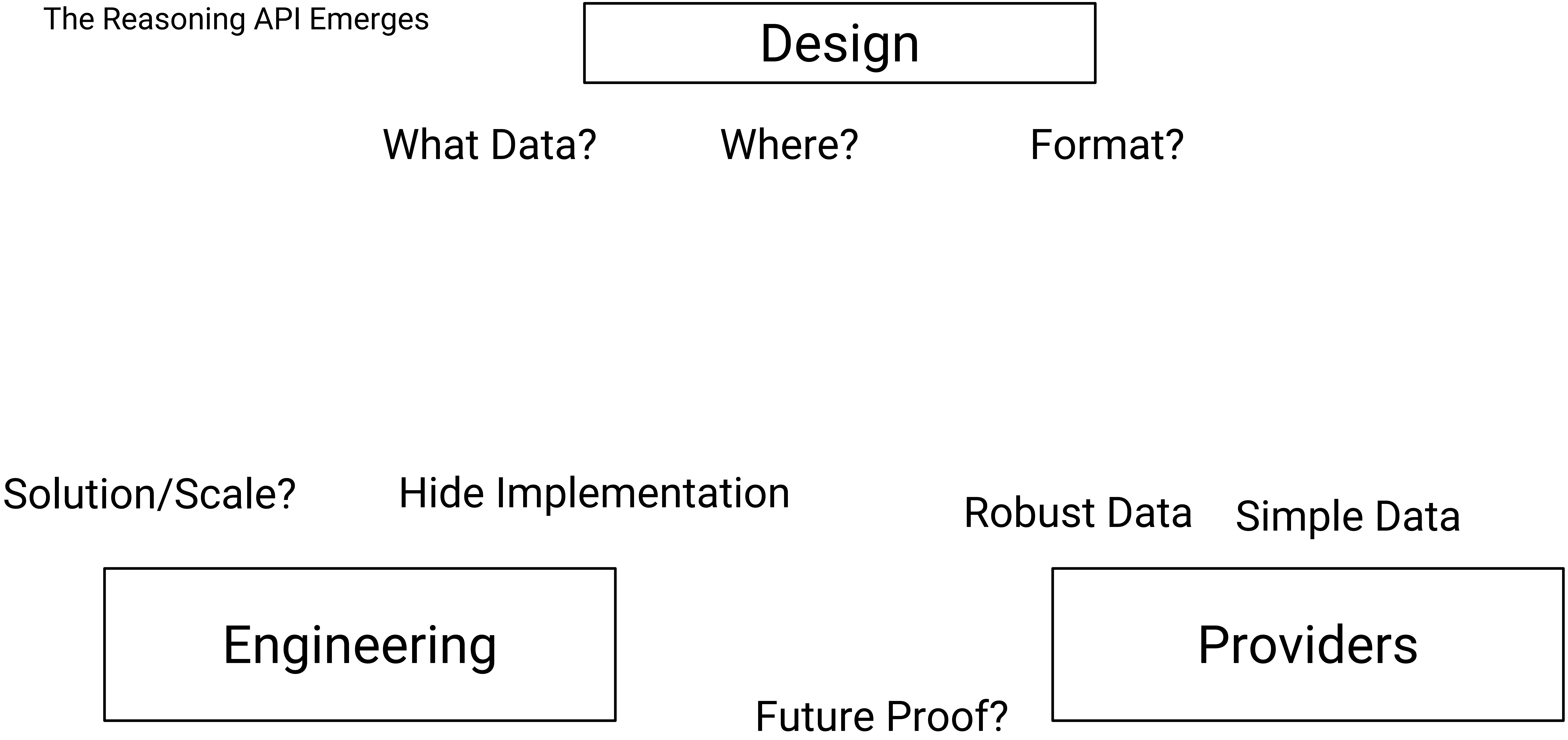


Design

Engineering

Providers

The Reasoning API Emerges



Design

What Data?

Where?

Format?

Solution/Scale?

Hide Implementation

Robust Data

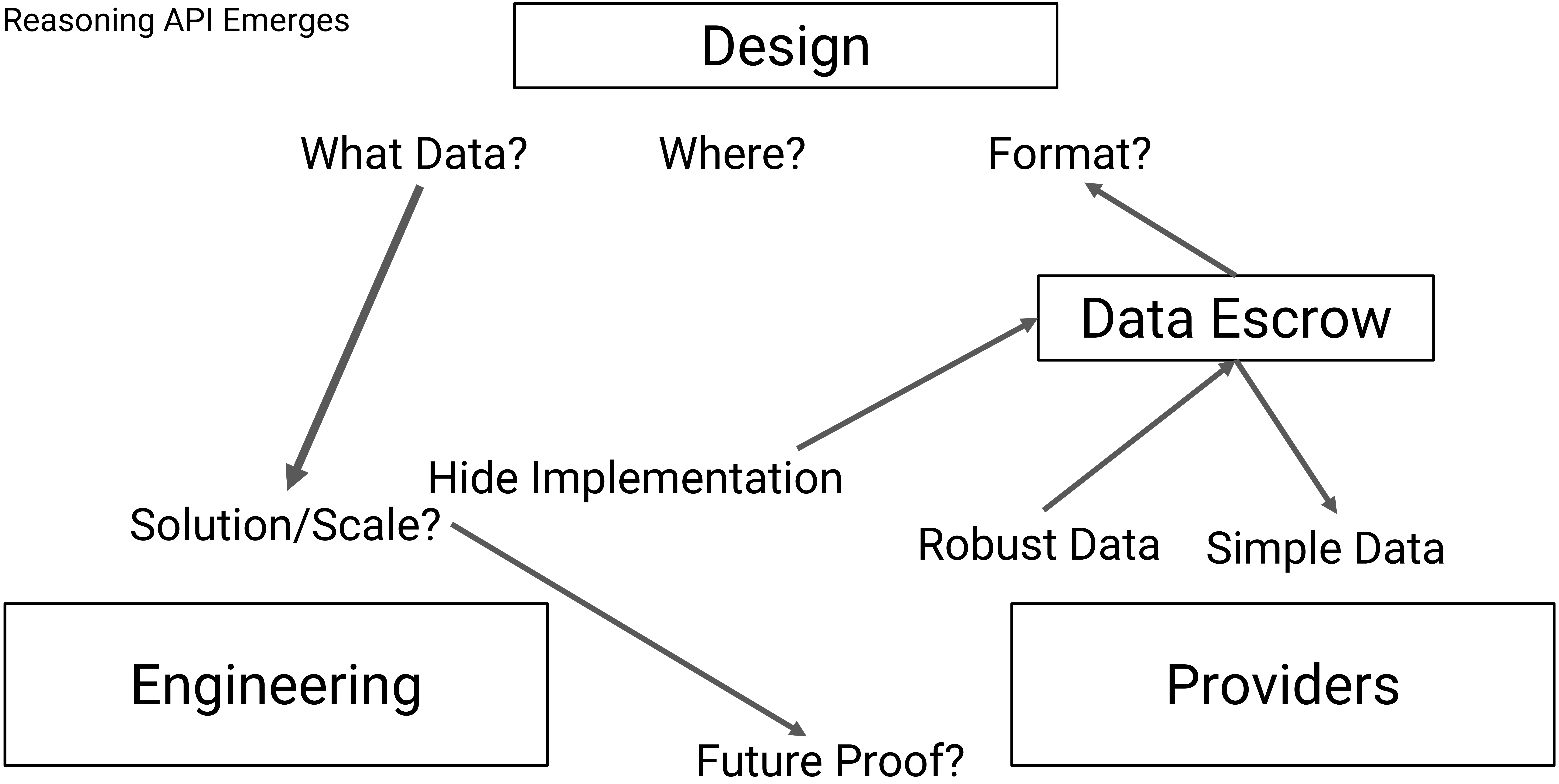
Simple Data

Engineering

Providers

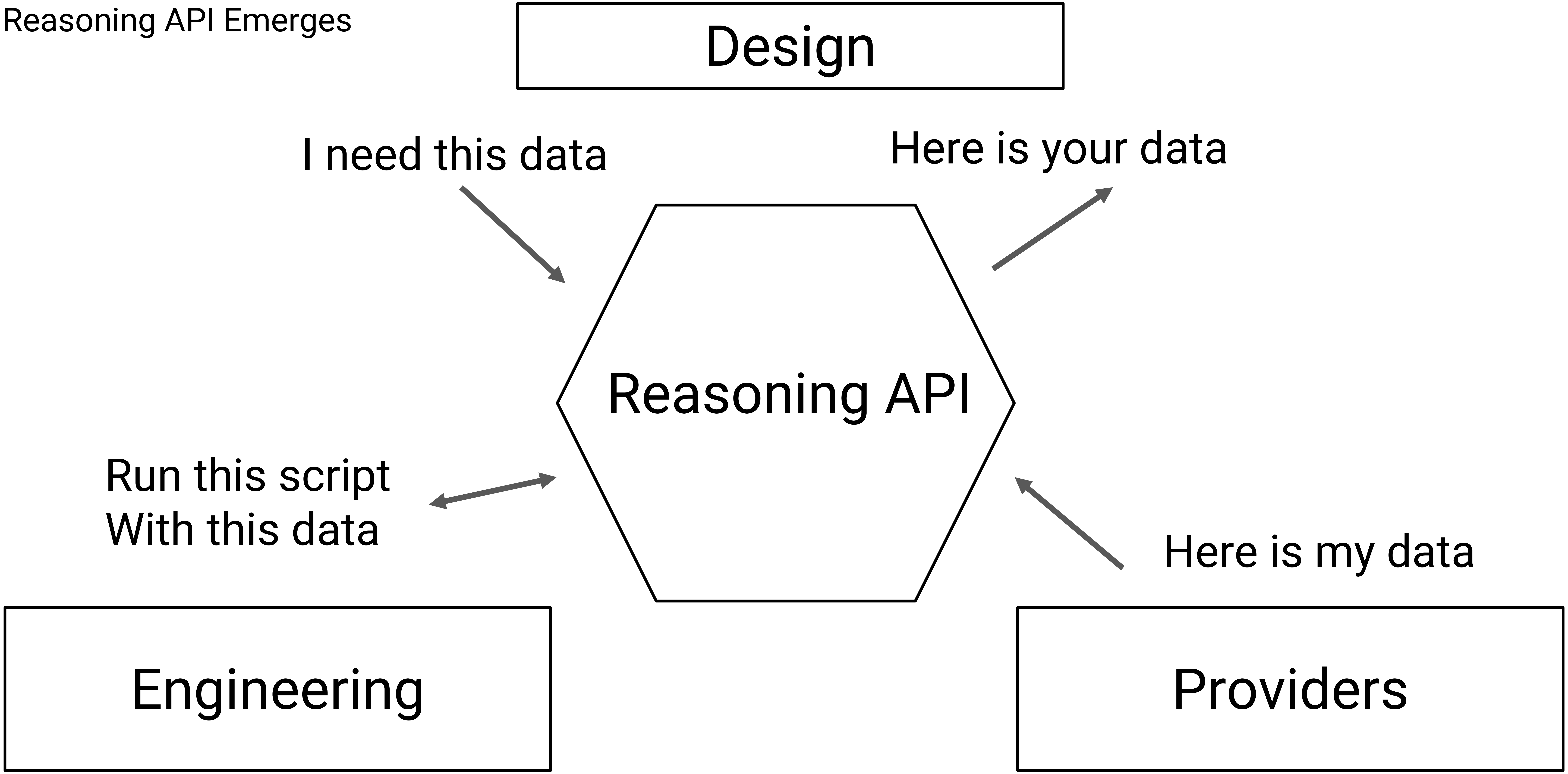
Future Proof?

The Reasoning API Emerges



unity

The Reasoning API Emerges



Design

I need this data

Here is your data

Reasoning API

Run this script
With this data

Here is my data

Engineering

Providers

The Reasoning API Emerges

Reasoning API

- ➡ Lets an app know what data it needs
- ⬅ Lets an app know what data it provides
- ↻ Operates on abstract ‘compatible’ data
- ↻ Creates or mutates data
- ⋗ Correlates unrelated data sets together
- ▨ Can be stacked on top of one another
- ❓ Occupies a grey area between provider and solution script

The Universal Data Layer

- ☑ *Not* a Reasoning API, but required
- ✎ Abstraction for design authoring
- Common language for Reasoning APIs and Providers
- Consists of
  - Primitive types: Numbers, characters, strings, Vectors
  - Semantic Tags: Strings (or strings encoded to IDs)
  - Rich data: Textures, models, animation, sound
    - Semantic tag is minimum requirement

When is Reasoning API used?

+Authored scene has *need* for it

Scene is authored with tagged data, implicitly have list

- No single Provider match

Available Providers + Scene requirements = Valid selection

No Available Reasoning API?

- App can't run

+Big hint to an engineer to write one!

tldr;

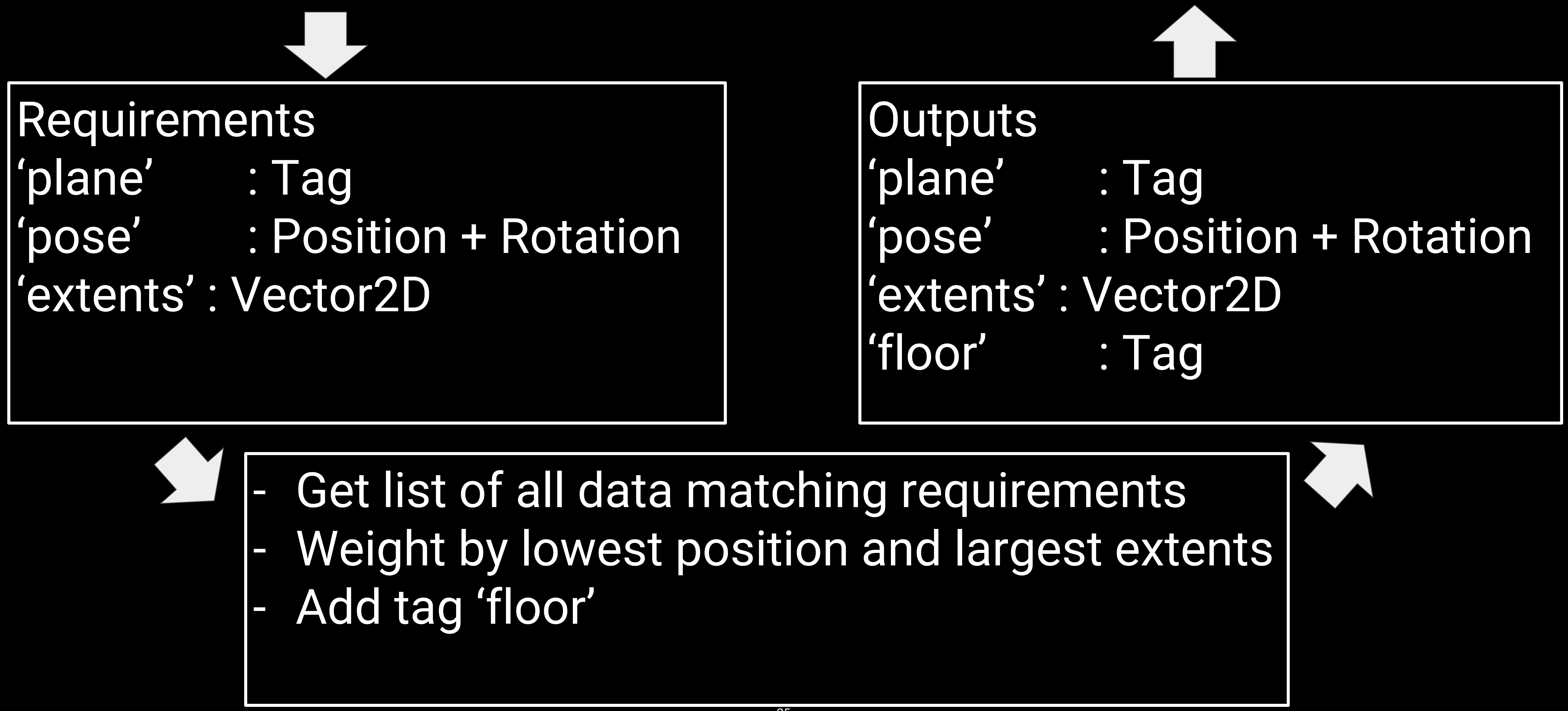
You have this big ‘data soup’

There are end pieces of data that are needed

Must pathfind to get from data-to-data-to-end condition

Reasoning APIs form connections along the data in the soup

Use Case 1 - ‘The floor’



Requirements
‘plane’ : Tag
‘pose’ : Position + Rotation
‘extents’ : Vector2D

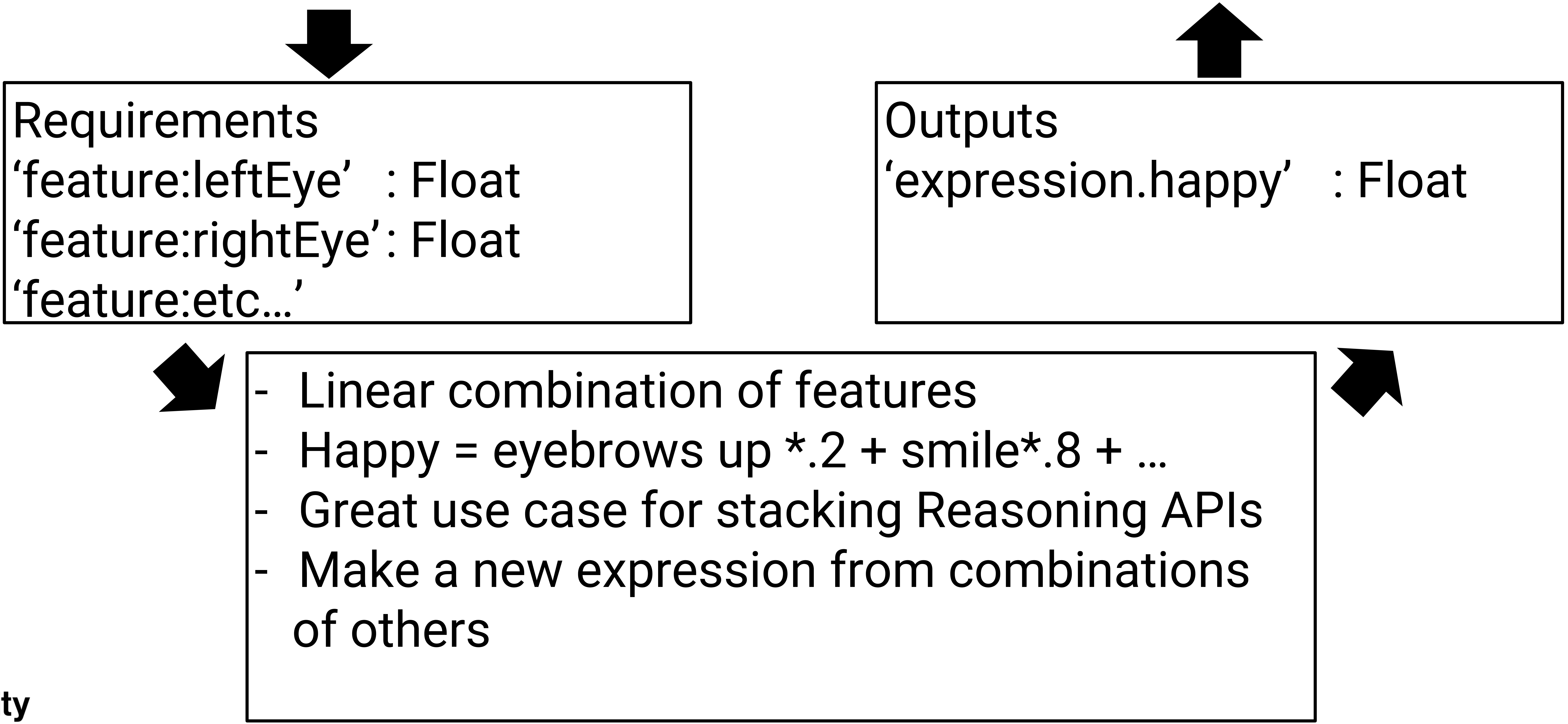
- Get list of all data matching requirements
- Weight by lowest position and largest extents
- Add tag ‘floor’

Outputs
‘plane’ : Tag
‘pose’ : Position + Rotation
‘extents’ : Vector2D
‘floor’ : Tag

The *Real* Challenge of AR

View example video at: https://youtu.be/BWMdlDN4OpU

Use Case 2 - Expression Calculus

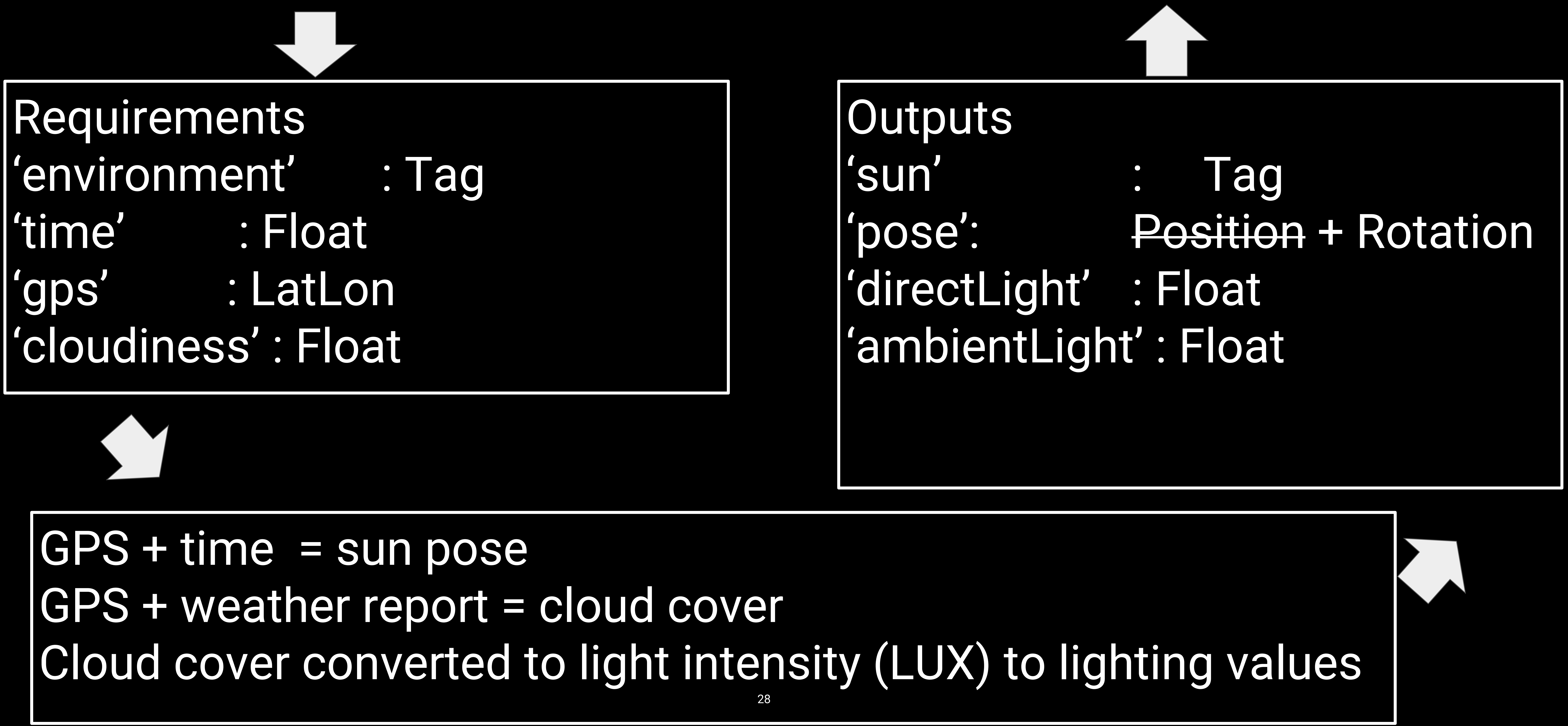


Requirements
‘feature:leftEye’ : Float
‘feature:rightEye’: Float
‘feature:etc…’

Outputs
‘expression.happy’ : Float

- Linear combination of features
- Happy = eyebrows up *.2 + smile*.8 + …
- Great use case for stacking Reasoning APIs
- Make a new expression from combinations of others

Use Case 3 - Weather to Lighting



Requirements

‘environment’ : Tag

‘time’ : Float

‘gps’ : LatLon

‘cloudiness’ : Float

Outputs

‘sun’ : Tag

‘pose’: ~~Position~~ + Rotation

‘directLight’ : Float

‘ambientLight’ : Float

GPS + time = sun pose

GPS + weather report = cloud cover

Cloud cover converted to light intensity (LUX) to lighting values

unity