

**CODE COFFEE**

**Augmented Reality  
for Your Posters**

# What is Augmented Reality?

## Augmented reality

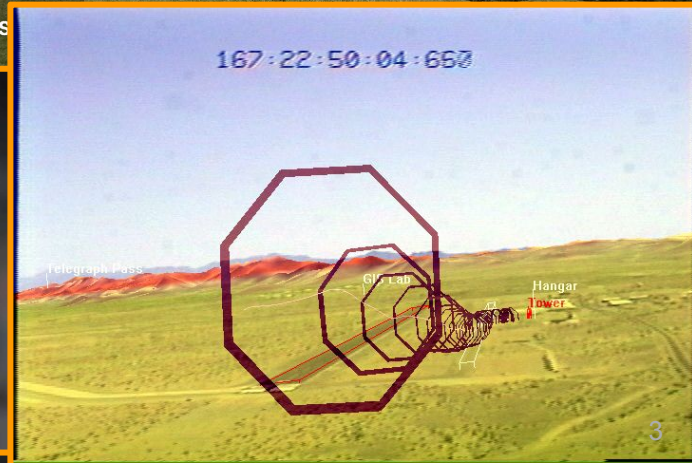
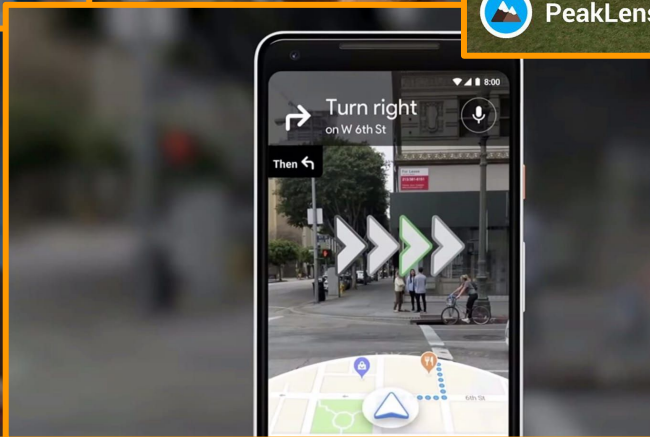
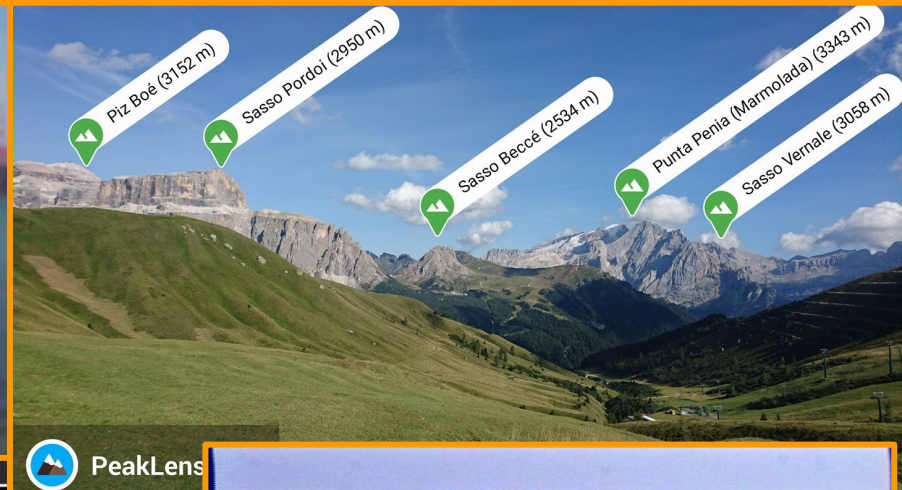
[Article](#) [Talk](#)

From Wikipedia, the free encyclopedia

*Not to be confused with [Virtual reality](#) or [Alternate reality](#).*

**Augmented reality (AR)** is an interactive experience that combines the real world and computer-generated content. The content can span multiple sensory [modalities](#), including [visual](#), [auditory](#),

# What is Augmented Reality?



# What is AR.js?

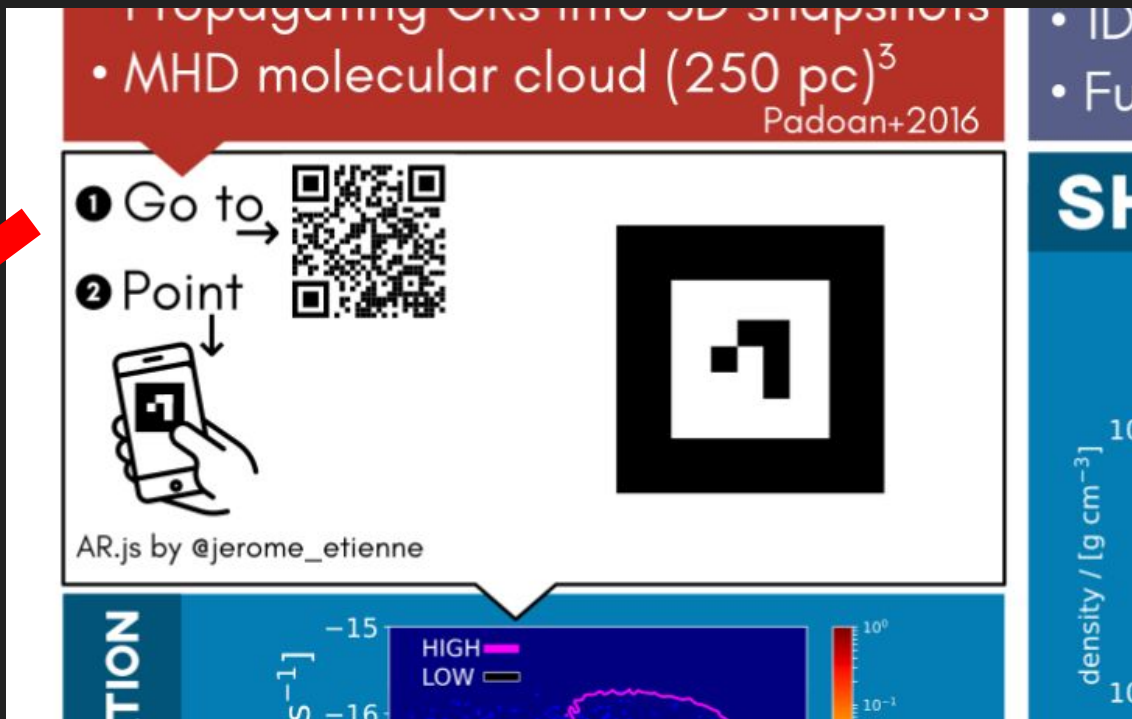
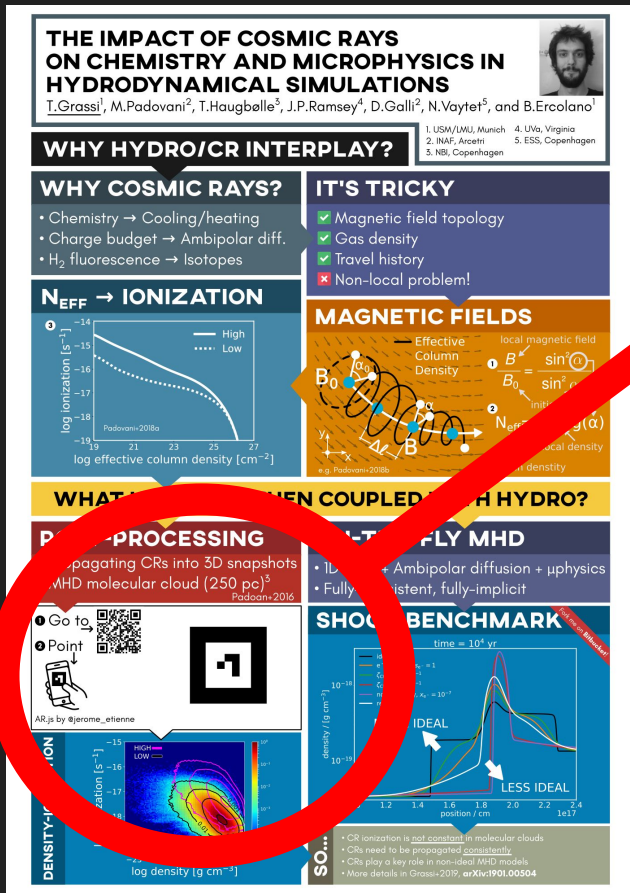


[ar-js-org.github.io/AR.js-Docs/](https://ar-js-org.github.io/AR.js-Docs/)

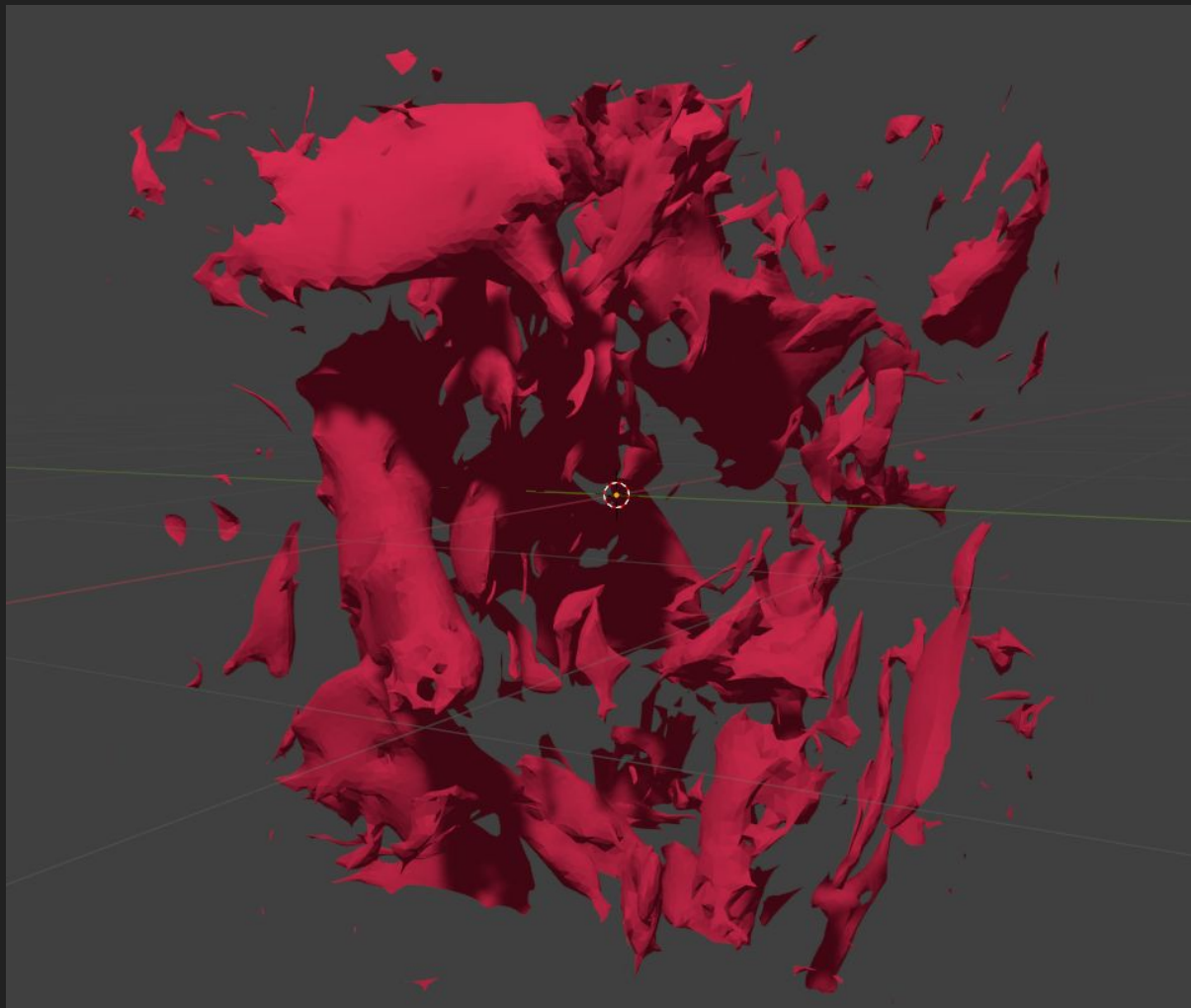
AR.js is a lightweight library for Augmented Reality on the Web, which includes features like:

- Image Tracking
  - Location based AR
  - Marker tracking
- 
- Very Fast: runs efficiently even on phones
  - Web-based: pure web solution, so no installation required. Fully JS based, using three.js + A-Frame + jsartoolkit5
  - Standards: works on any phone with webgl and webrtc
  - Open Source

# Motivation: 3D visualization in posters





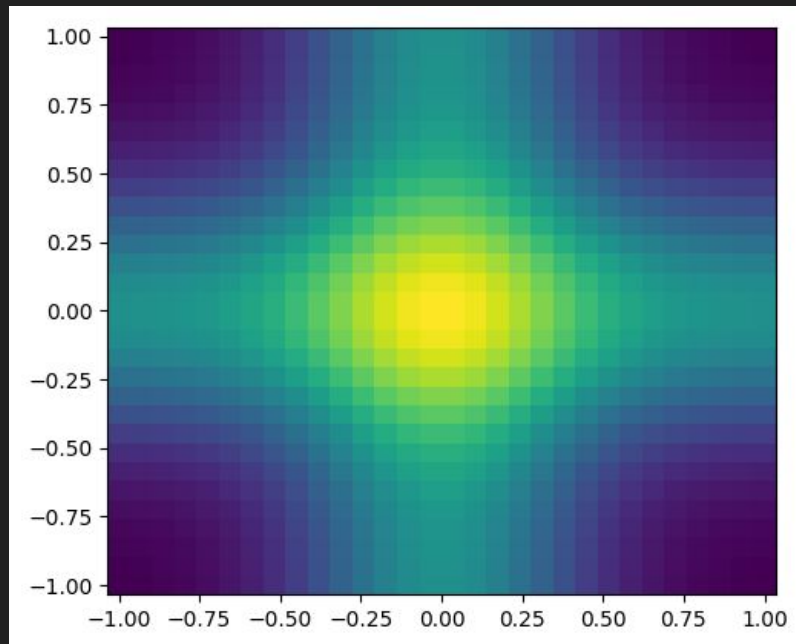


# Real Motivation



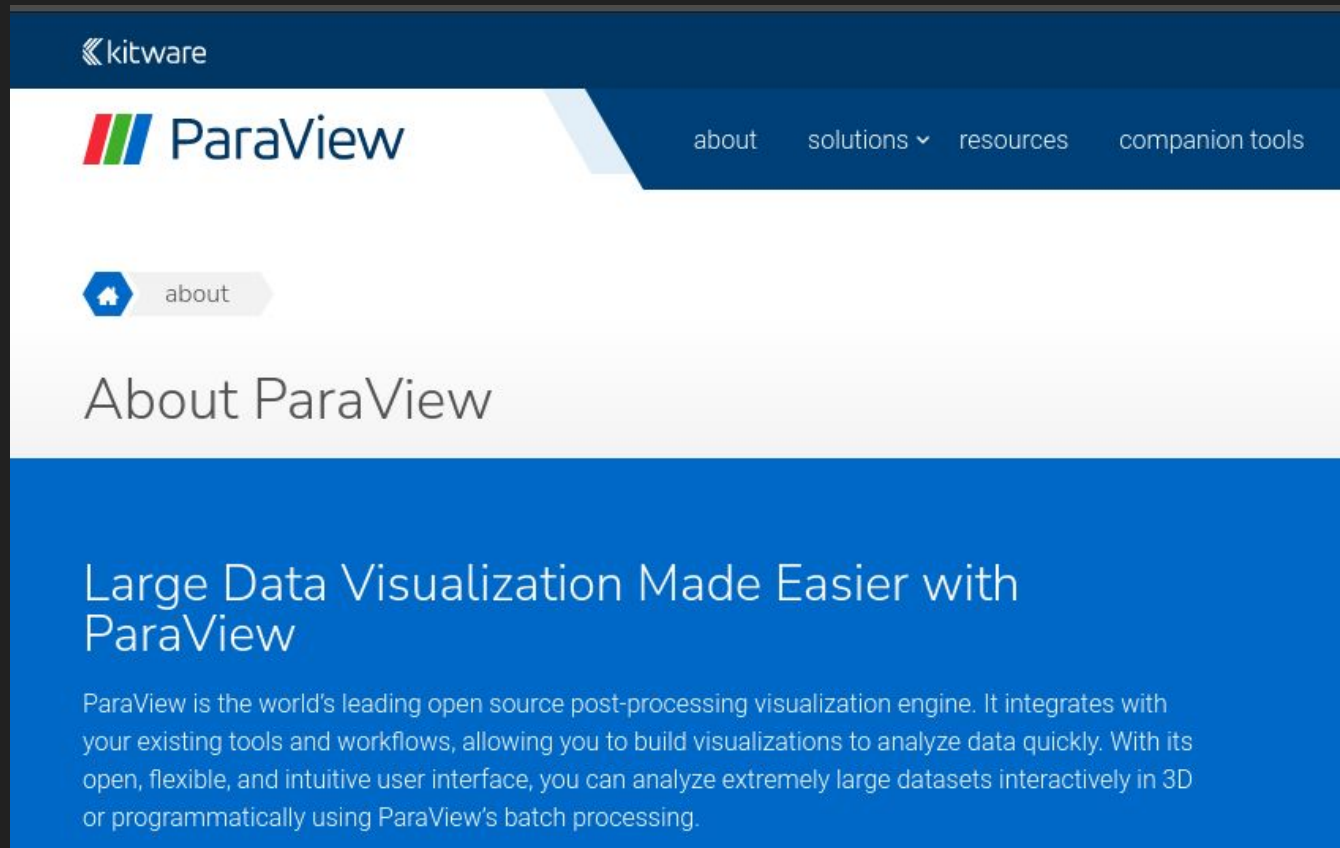
# Step 1: produce some 3D data

```
main.py x
1 import matplotlib.pyplot as plt
2 import numpy as np
3
4 # number of grid points per side
5 n = 30
6
7 # create grids
8 x = np.linspace(-1, 1, n)
9 y = np.linspace(-1, 1, n)
10 z = np.linspace(-1, 1, n)
11
12 # create meshgrid
13 xx, yy, zz = np.meshgrid(x, y, z, indexing="ij")
14
15 # create some 3D function
16 sx = sy = sz = 0.1
17 ff = np.exp(-xx**2 / 2 / sx) + np.exp(-yy**2 / 2 / sy) + np.exp(-zz**2 / 2 / sz)
18
19 # plot slice to test
20 plt.pcolormesh(xx[..., n//2], yy[..., n//2], ff[..., n//2])
21 plt.show()
22
23 # prepare for csv file
24 data = np.array([np.ravel(x) for x in [xx, yy, zz, ff]]).T
25
26 # save to csv file
27 np.savetxt("data.csv", data)
```





# Step 2: produce a 3D object file



The image is a screenshot of the ParaView website. At the top, there is a dark blue header with the Kitware logo on the left. Below the header, the ParaView logo is prominently displayed on the left, and a navigation menu with links for 'about', 'solutions', 'resources', and 'companion tools' is on the right. A secondary navigation bar below the header features a home icon and the word 'about'. The main content area has a white background with the heading 'About ParaView'. Below this, a large blue section contains the text 'Large Data Visualization Made Easier with ParaView' and a paragraph describing ParaView as an open-source post-processing visualization engine.

kitware

ParaView

about solutions ▼ resources companion tools

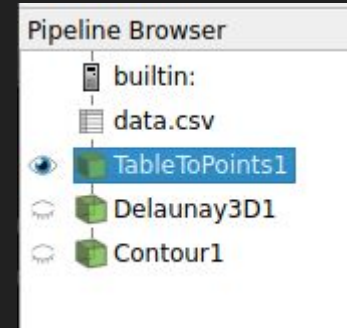
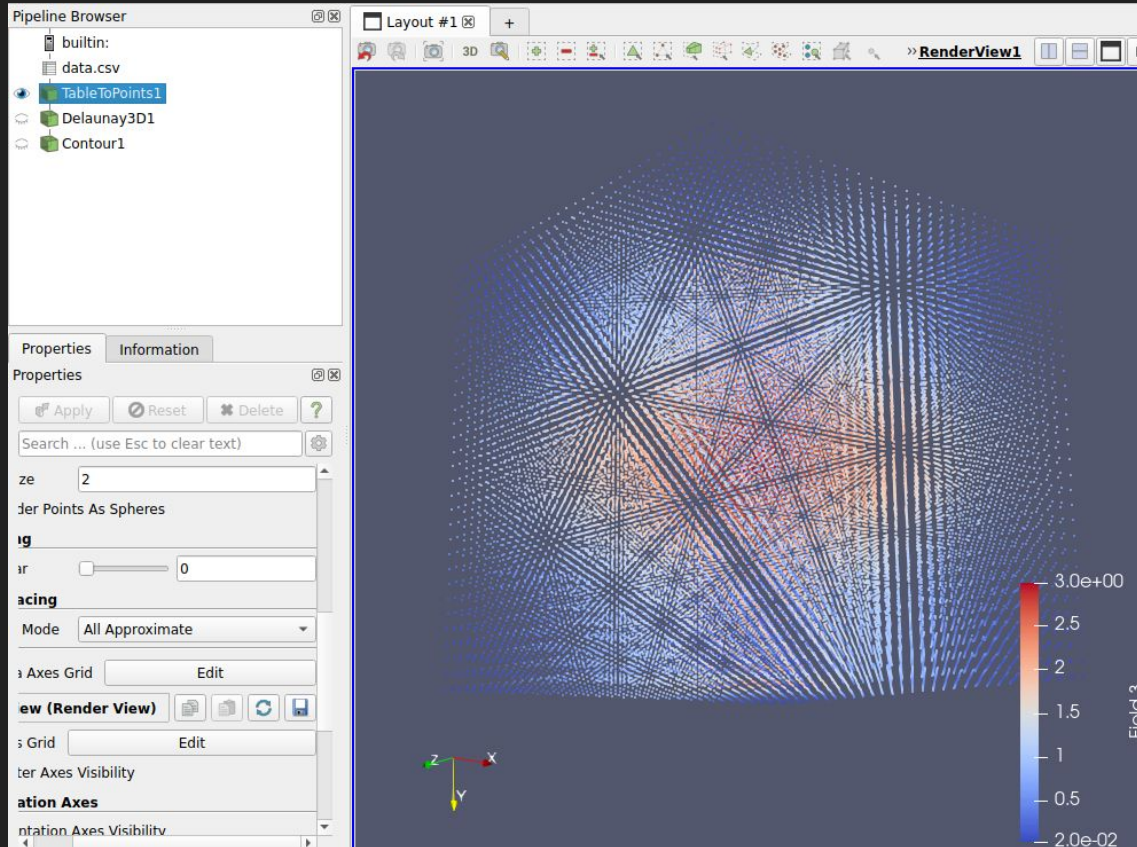
about

## About ParaView

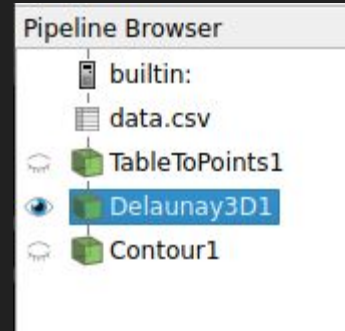
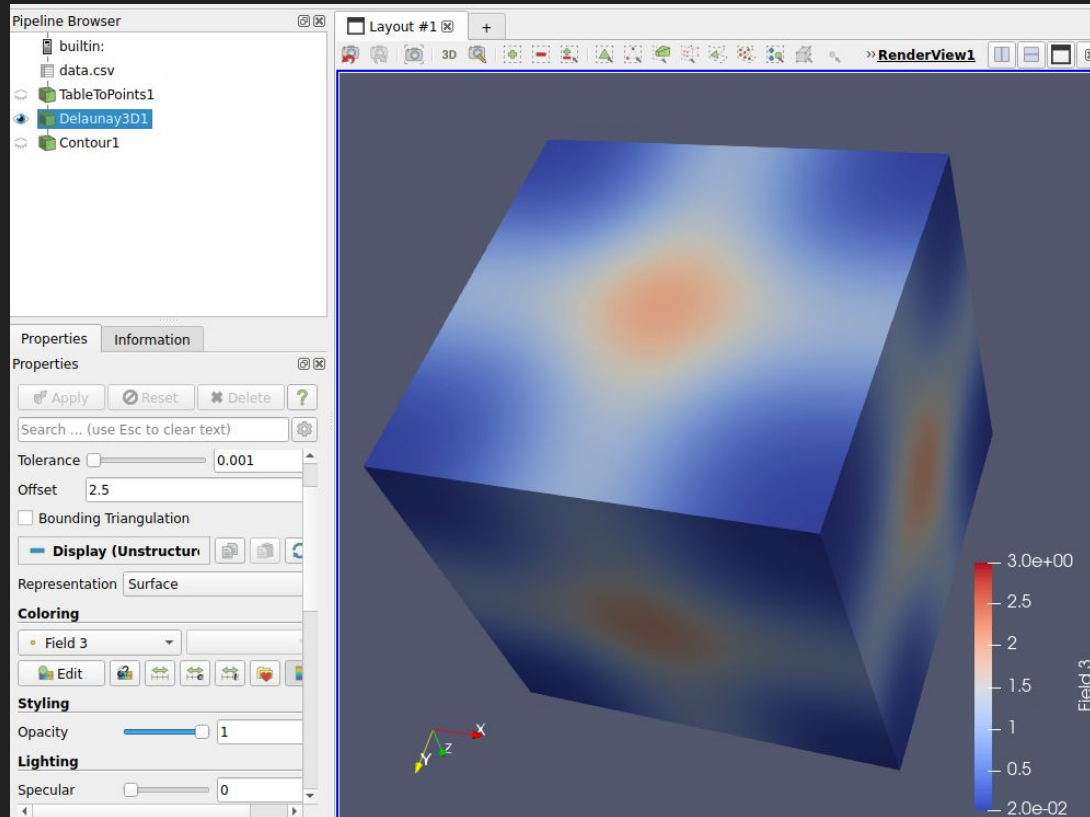
### Large Data Visualization Made Easier with ParaView

ParaView is the world's leading open source post-processing visualization engine. It integrates with your existing tools and workflows, allowing you to build visualizations to analyze data quickly. With its open, flexible, and intuitive user interface, you can analyze extremely large datasets interactively in 3D or programmatically using ParaView's batch processing.

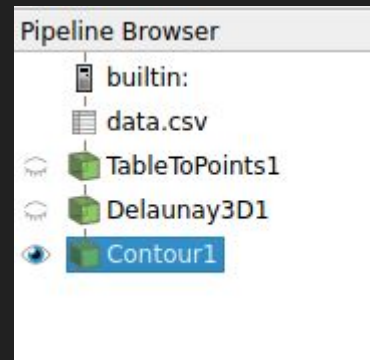
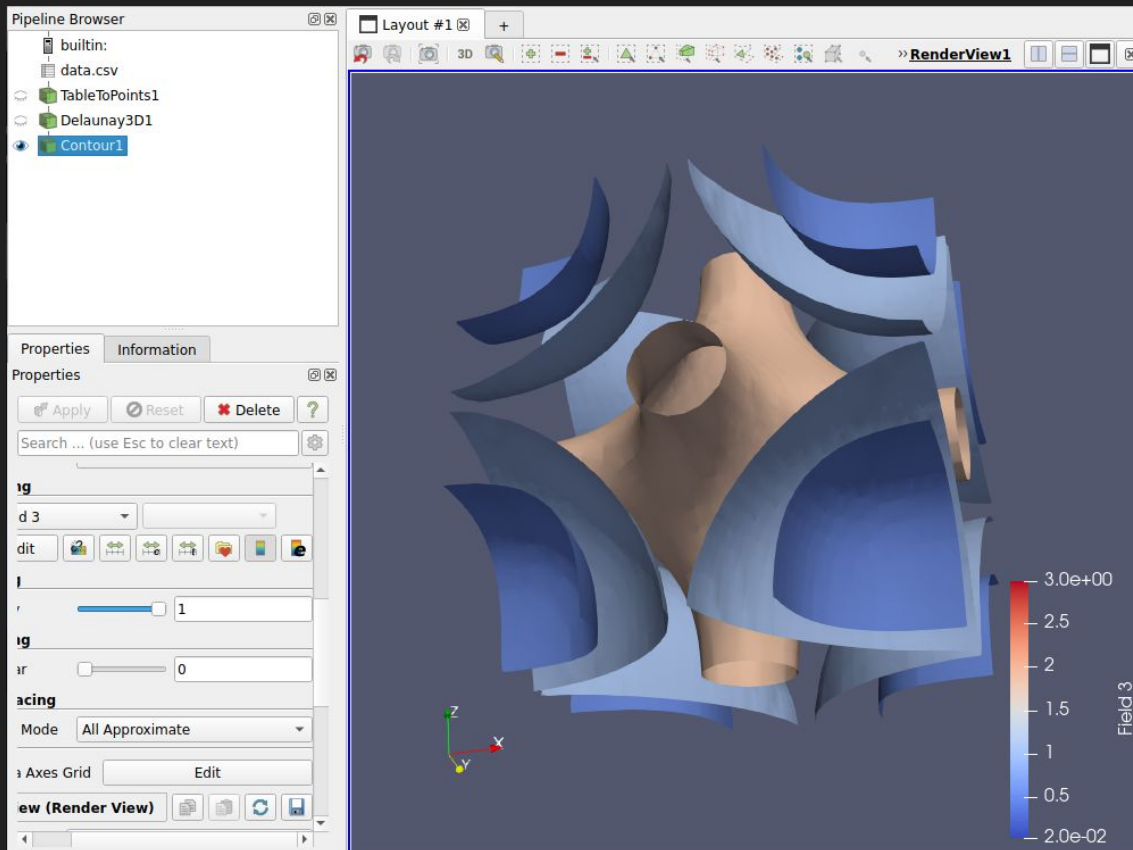
# Step 2: produce a 3D object file



# Step 2: produce a 3D object file



# Step 2: produce a 3D object file



Export Scene -> \*.gltf

# Before the next step: what is A-FRAME?

## A-FRAME

- DOCS
- BLOG
- COMMUNITY
- SHOWCASE
- GITHUB
- SLACK
- DISCORD
- NEWSLETTER
- ASK A QUESTION

VERSION 1.4.0

- INTRODUCTION
- Introduction
- Installation
- VR Headsets & WebXR Browsers
- HTML & Primitives

1.4.0 > INTRODUCTION

## Introduction

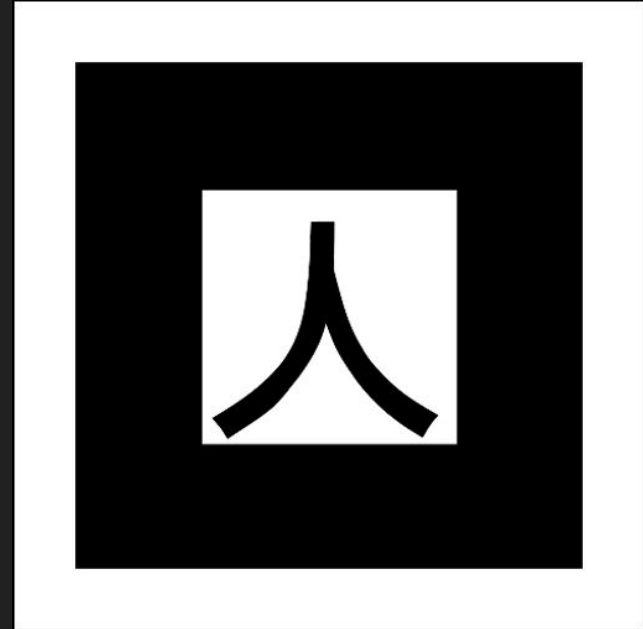
### Getting Started

A-Frame can be developed from a plain HTML file without having to install anything. A great way to try out A-Frame is to [remix the starter example on Glitch](#), an online code editor that instantly hosts and deploys for free. Alternatively, create an `.html` file and include A-Frame in the `<head>`:

```
<html>
  <head>
    <script src="https://aframe.io/releases/1.4.0/aframe.min.js"></script>
  </head>
  <body>
    <a-scene>
      <a-box position="-1 0.5 -3" rotation="0 45 0" color="#4CC3D9"></a-box>
      <a-sphere position="0 1.25 -5" radius="1.25" color="#EF2D5E"></a-sphere>
      <a-cylinder position="1 0.75 -3" radius="0.5" height="1.5" color="#4CC3D9"></a-cylinder>
      <a-plane position="0 0 -4" rotation="-90 0 0" width="4" height="4" color="#ECECEC"></a-plane>
      <a-sky color="#ECECEC"></a-sky>
    </a-scene>
  </body>
</html>
```



# Before next step: what is a marker?



## Before the next step: what is a marker?

# AR.js Marker Training

Pattern Ratio 0.54

Image size 620px

Border color. Please choose a dark one.

black

UPLOAD

DOWNLOAD MARKER

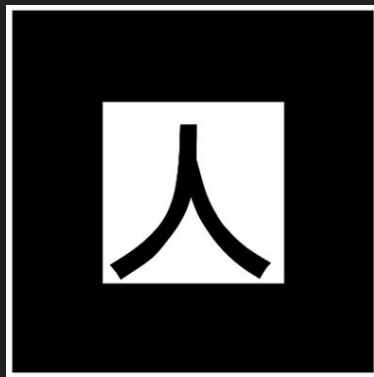
PDF ONE PER PAGE

DOWNLOAD IMAGE

PDF TWO PER PAGE

PDF SIX PER PAGE

AR.js



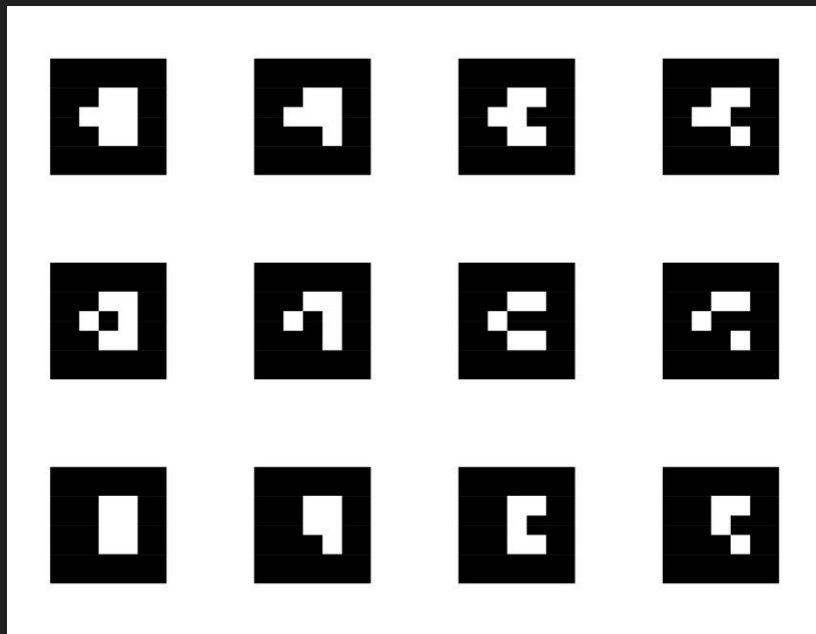
# .patt file

[illegible]

# Before the next step: what is a marker?

Some tips:

- Avoid overly complicated patterns
- Use rotationally asymmetrical markers
- From portrait to landscape and back
- Marker size
- Proportional scale
- Print the marker
- Proper border size around the marker
- Chrome on iOS: we have a problem
- Change camera resolution



# Step 3: Prepare JS code (actually just HTML)

index.html

```
1 <!DOCTYPE html>
2 <html>
3   <script src="js/aframe.min.js"></script>
4   <!-- we import arjs version without NFT but with marker -
5   <script src="js/aframe-ar.js"></script>
6   <body style="margin : 0px; overflow: hidden;">
7     <a-scene embedded arjs>
8       <a-marker type='pattern' url='kanji.patt'>
9         <!-- we use cors proxy to avoid cross-origin pro
10        <a-entity
11          position="0 0 0"
12          scale="1 1 1"
13          gltf-model="models/testx.gltf"
14        ></a-entity>
15      </a-marker>
16      <a-entity camera></a-entity>
17    </a-scene>
18  </body>
19 </html>
```

import necessary JS libraries

embed in a a-scene

define marker

assign gltf model to marker

add a camera

# Step 4: Load on some server

## YOUR SERVER



important: https



## Step 5: Test it!



marker on slide 14

# Final remarks

- Use relatively light 3D models (it has to be downloaded)
- Animations can be used, but with Paraview is complicated
- A-frame animations are available (e.g. constant rotation)
- VR goggles
- Note that sometimes Paraview produces “wrong” gltf models (solved with x3d and Blender)
- Scaling is important
- Light conditions are important (reflections on black prints)
- Fancy but not impressive
- A-Frame allows interactions and triggered events
- It is possible to use textures and advanced lighting
- Also location-based and image-based tracking is available