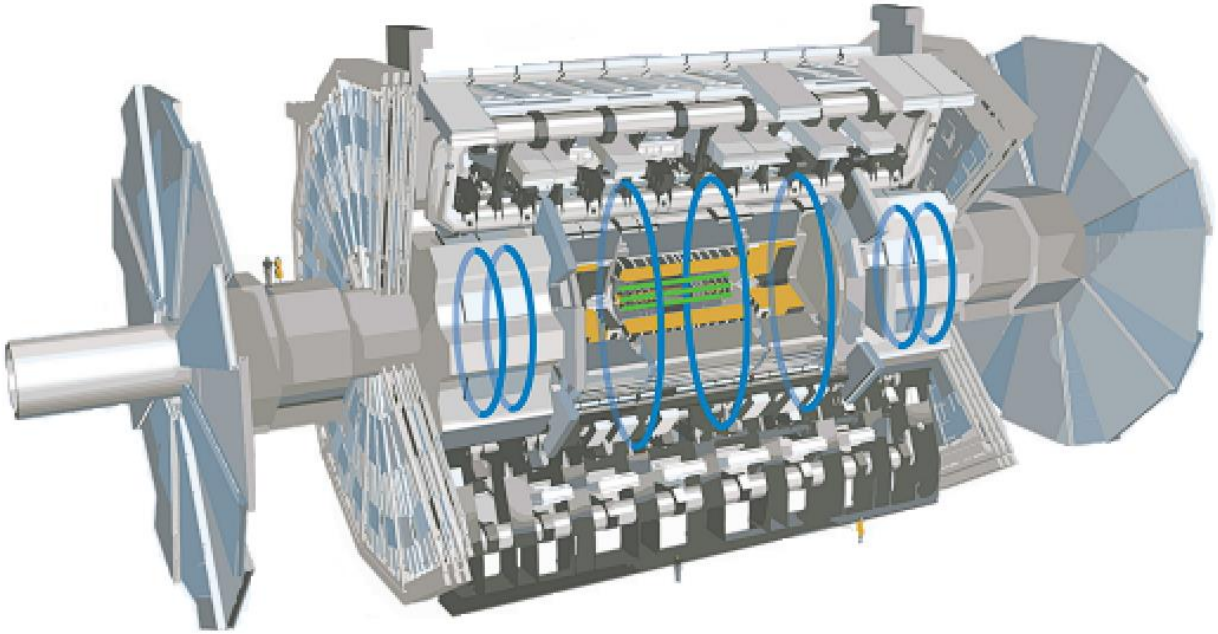
AR BOOK

Project to demonstrate a 3D Scene in AR Foundation



Research and Development stage presentation

Draft presentation with QR code scanner

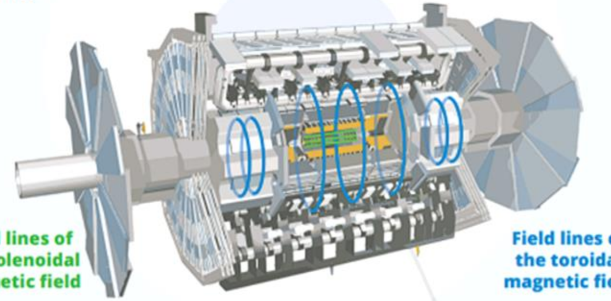
ATLAS-GTU TAI Agreement Workshop

Georgian Team: Vladimir Dolinski
Responsible: Lasha Sharmazanashvili

PROJECT HIGHLIGHT

MAGNET SYSTEM

ATLAS uses two different types of superconducting magnet systems – solenoidal and toroidal. When cooled to about 4.5 K (–268°C), these are able to provide strong magnetic fields that bend the trajectories of charged particles. This allows physicists to measure their momentum and charge.



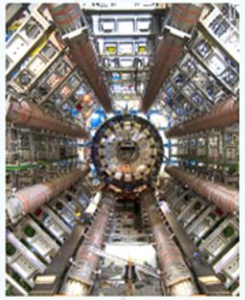
CENTRAL SOLENOID MAGNET

The ATLAS solenoid surrounds the inner detector at the core of the experiment. This powerful magnet is 5.6 m long, 2.56 m in diameter and weighs over 5 tonnes. **It provides a 2 Tesla magnetic field in just 4.5 cm thickness.** This is achieved by embedding over 9 km of niobium-titanium superconductor wires into strengthened, pure aluminum strips, thus minimising possible interactions between the magnet and the particles being studied.

TOROID MAGNET

The ATLAS toroids use a series of eight coils to provide a magnetic field of up to 3.5 Tesla, used to measure the momentum of muons. There are **three toroid magnets** in ATLAS: two at the ends of the experiment, and one massive toroid surrounding the centre of the experiment.

At 25.3 m in length, the central toroid is the **largest toroidal magnet ever constructed** and is an iconic element of ATLAS. It uses over 56 km of superconducting wire and weighs about 830 tonnes. The end-cap toroids extend the magnetic field to particles leaving the detector close to the beam pipe. Each end-cap is 10.7 m in diameter and weighs 240 tonnes.



https://atlas.cern

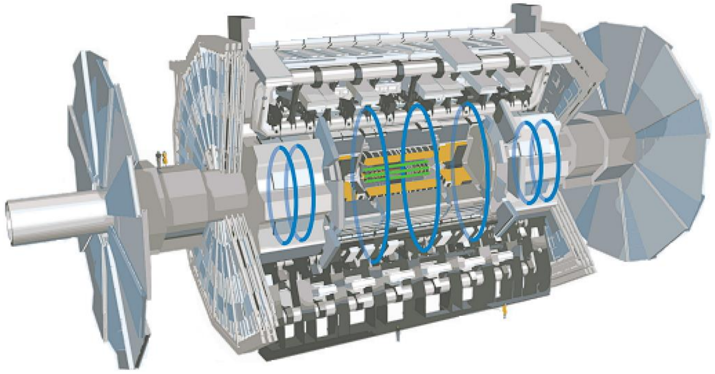
ATLAS EXPERIMENT

AGENDA

1. Main purpose
2. Project basis
3. Problem
4. Solution
5. Roadmap
6. Workspace
7. Draft presentation

MAIN PURPOSE

SHOW ATLAS DETECTOR GEOMETRY IN AR

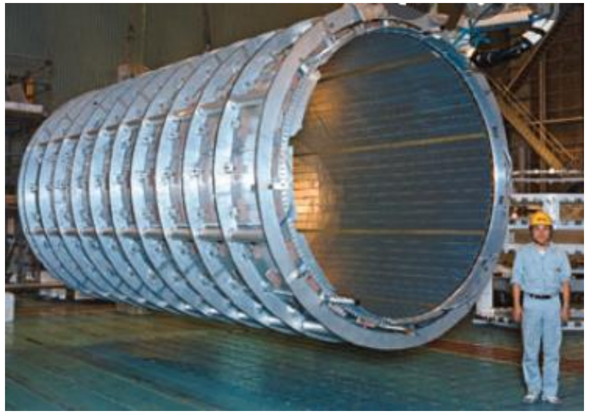


We need to show:

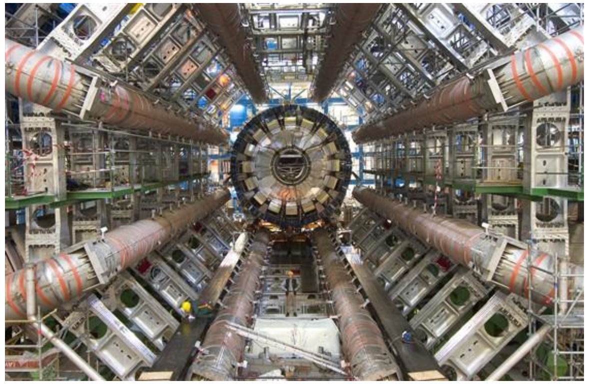
1. Geometry
2. Scale
3. Magnetic fields

of Central solenoid and Toroid magnet
And application must be controllable

CENTRAL SOLENOID

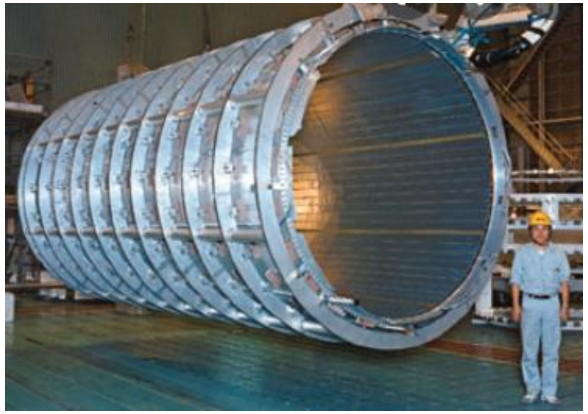


TOROID MAGNET



MAIN PURPOSE

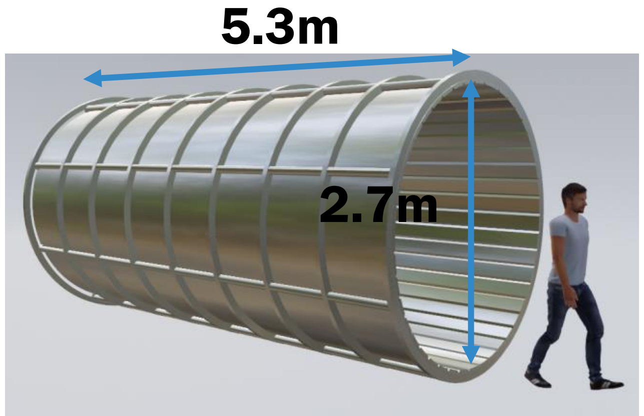
CENTRAL SOLENOID SECTION



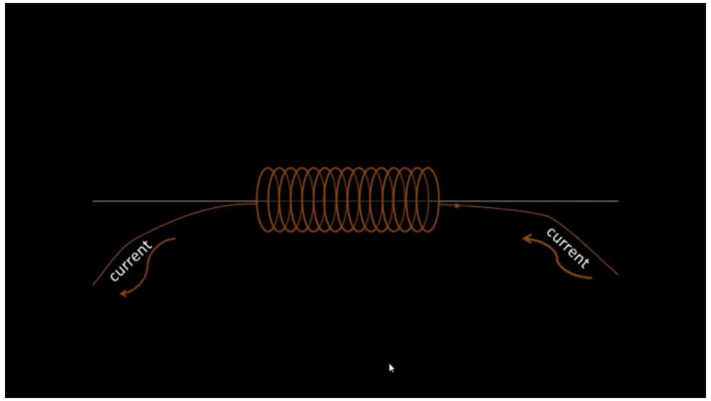
Scene with Central solenoid contains:

1. geometry (Hight 2.7m, Length 5.3m)
2. human for scale (Hight 1.8m)
3. magnetic field

GEOMETRY AND SCALE

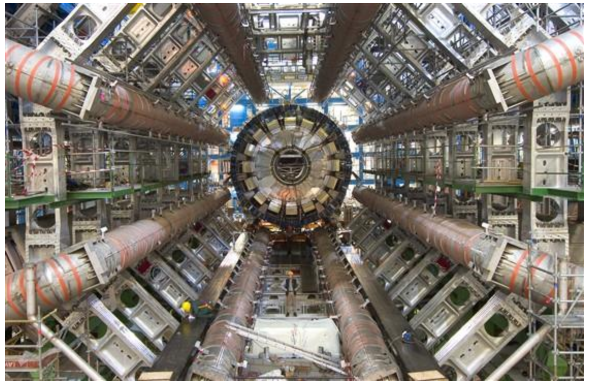


MAGNETIC FIELD

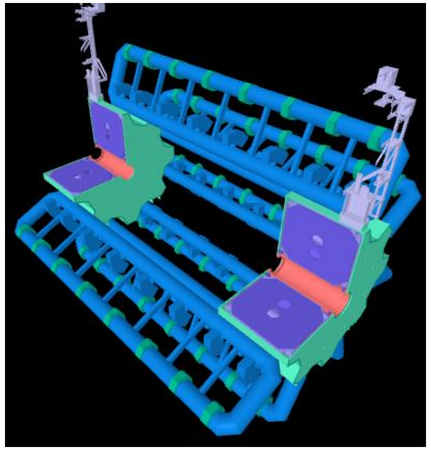


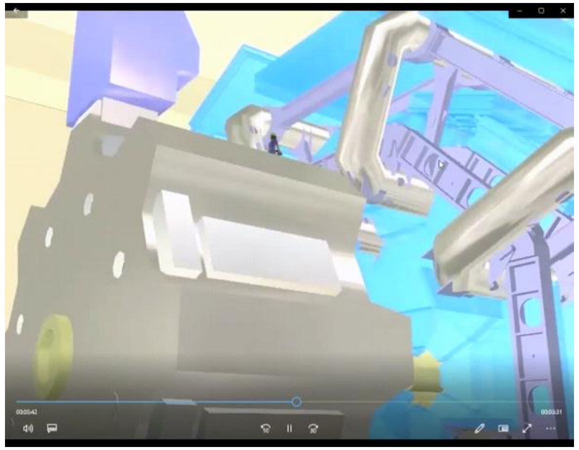
MAIN PURPOSE

TOROID MAGNET SECTION

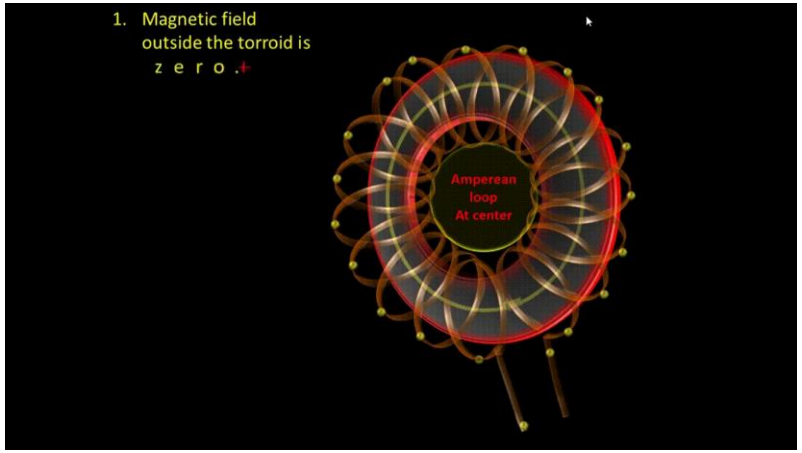


GEOMETRY AND SCALE





MAGNETIC FIELD



Scene with Toroid magnet contains:

1. geometry (Barrel and Endcap)
2. human for scale
3. magnetic field

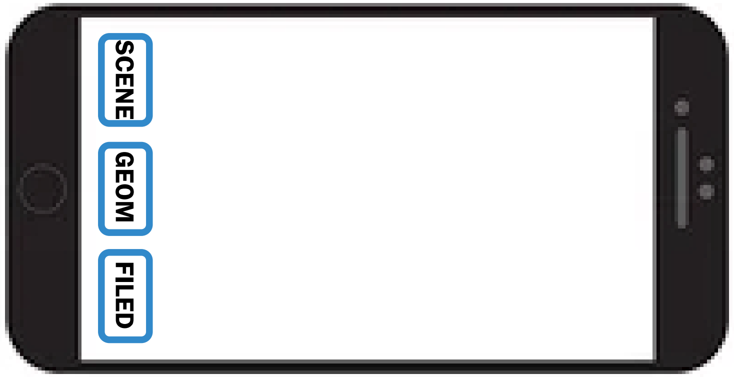
MAIN PURPOSE

BACKEND SECTION

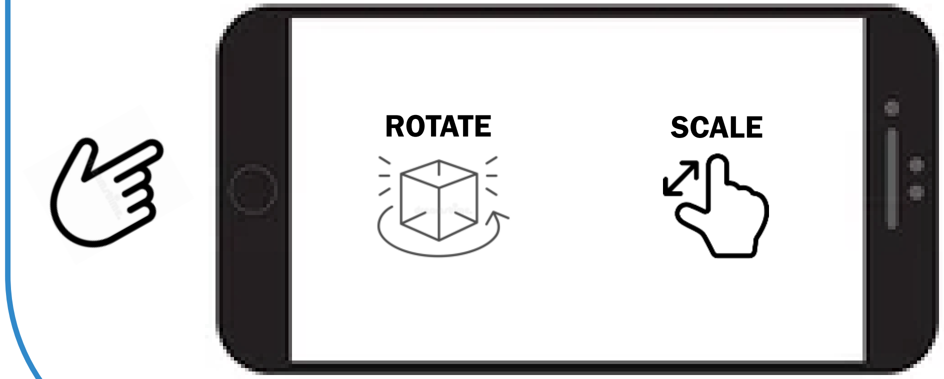
1. WHOLE SCENE HIDE/SHOW
2. ONLY GEOMETRY HIDE/SHOW
3. MAGNETIC FIELD HIDE/SHOW

APPLICATION MUST BE CONTROLLABLE
BOTH WITH BUTTONS AND GESTURE CONTROLS

BUTTON-BOUND FUNCTIONS



GESTURE CONTROL FUNCTIONS



PROJECT BASIS

Research

I. understand what AR is

II. finding a way for the application to work

III. find out the requirements

IV. find tools with which we can create an application

V. define pros and cons of the chosen decisions

Development

A. visualization stage:
   I. geometry creation
   II. adding animation
   III. exporting scene

B. coding stage:
   I. adding image targeting
   II. importing scene
   III. code writing

PROBLEM

Research

WHAT IS AR?

Augmented reality (AR) is an interactive experience that combines the real world and computer-generated content.

HOW I WORK WITH IT?

Our AR software works on JavaScript.

AR can be used in local area, can be image or marker targeted .

PROBLEM

Research

WHAT IS REQUIREMENTS?

For visualization I used Blender, it has all the necessary tools for working with geometry and animation and further export.

Other variants was 3ds max, unity.

PROS AND CONS

- Free
- Big community to find solutions
- All necessary tools in one package
- Some of them are outdated

PROBLEM

Research

WHAT IS REQUIREMENTS?

For JavaScript, we need a code editor and a server to build app publish it.

AR on JS itself have different frameworks:
AR.JS, WebXR, ARCore, Kudan, AppleAR and etc.

PROS AND CONS

- Open source
- Big variety of tools
- Works in browser
- Three.JS is less used nowadays

PROBLEM

Development

FRONTEND

Smarteam have geometry, but it's in WRL format, which is not AR friendly.

For magnetic field I needed to find balance between lines and particle count.

Human mesh was imported, but not rig, because tools are outdated.

Lighting in JS is not good enough.

BACKEND

To work with 3D I needed renderer.

To publish code I needed server.

Documentary is about what you can do, not how.

Code implementations founded on forums/other sites was outdated or not compatible.

SOLUTION

Development

FRONTEND

I created geometry from scratch in Blender.

With some iterations I stopped on not very dense field and approximately 100 balls(particles).

Rig for human done by myself.

Magnetic field and human were animated separately (can’t be grouped at this stage).

I imported HDRi scene with global lighting.

BACKEND

For renderer I choose Three.JS.

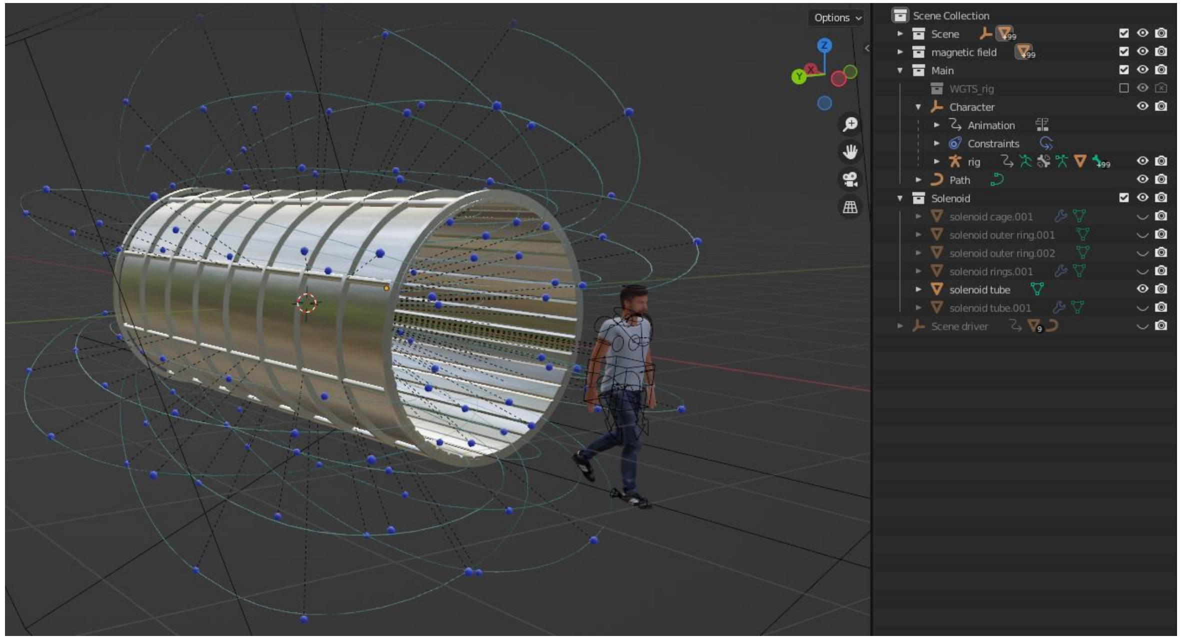
To publish I had variants to do it on GitHub server or 3rd party service. I choose second, because it’s fast and do not require skills.

I learned some JS for further work, but three.js should be replaced by another framework for fast and convenient work on the code.

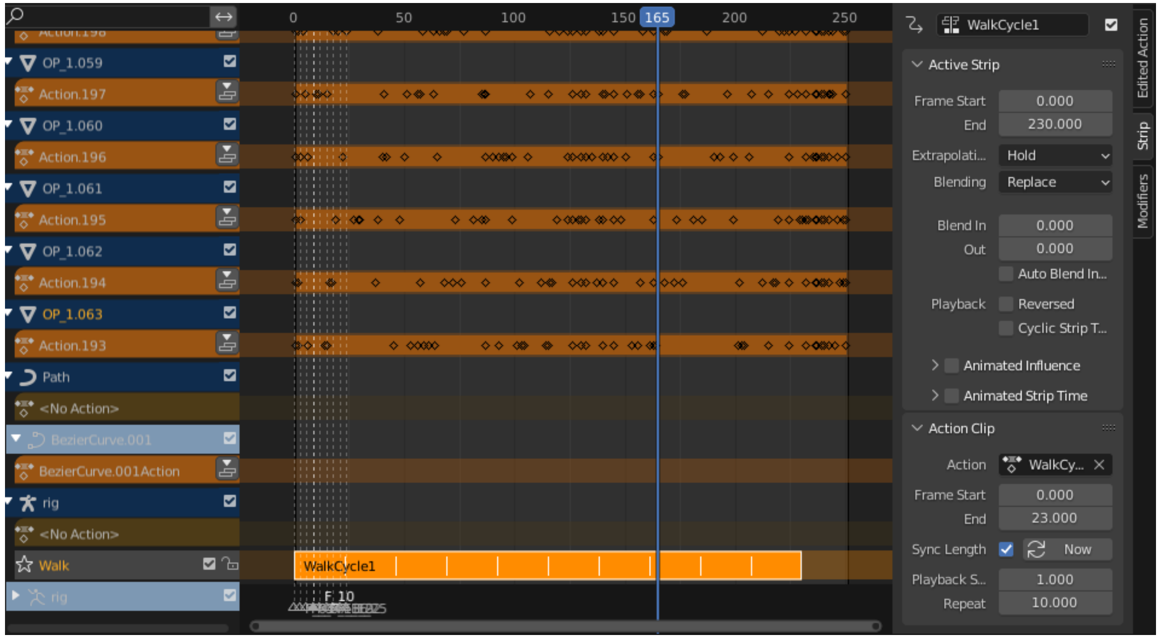
WORKSPACE

Frontend

Geometry



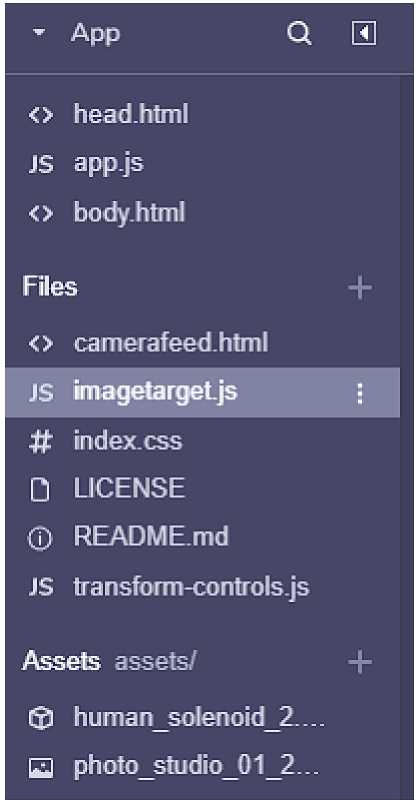
Animation



WORKSPACE

Backend

Project Tree



in head.html I have imported loaders

main code section

geometry and lighting

Code

1

```
// Populates some object into an XR scene and sets the initial camera position. The scene and
// camera come from xr3js, and are only available in the camera loop lifecycle onStart() or later.
const initXrScene = ({scene, camera, renderer}) => {
  // Enable shadows in the rednerer.
  renderer.shadowMap.enabled = true
  renderer.shadowMap.type = THREE.PCFSoftShadowMap
  // Add some light to the scene.
  renderer.physicallyCorrectLights = true
  renderer.toneMapping = THREE.ACESFilmicToneMapping
  renderer.toneMappingExposure = 0.4
  renderer.outputEncoding = THREE.sRGBEncoding
```

2

```
  const rgbeloader = new THREE.RGBELoader()
  rgbeloader.setDataType(THREE.UnsignedByteType)
  rgbeloader.load(HdrFile, (texture) => {
    texture.mapping = THREE.EquirectangularReflectionMapping
    scene.environment = texture
  })
```

3

```
  // Load 3D model
  loader.load(
    // resource URL
    modelFile,
    // loaded handler
    (gltf) => {
      model = gltf.scene
      model.castShadow = false
```

4

```
      // animate the model
      mixer = new THREE.AnimationMixer(model)
      const clip = gltf.animations[0]
      mixer.clipAction(clip.optimize()).play()
      animate()
      scene.add(model)
      // model.scale.set(0.2, 0.2, 0.2)
      // Hide 3D model until image target is detected.
      model.visible = false
    }
  )
```

ROADMAP

	COMPLETED	IN PROGRESS	IN DESIGN
FRONTEND	SOLENOID GEOMETRY	SOLENOID ENHANCEMENT	2D ICONS FOR APP
	HUMAN ANIMATION	ANIMATION ENHANCEMENT	TOROID GEOMETRY
	MAGNETIC FIELD GEOMETRY&ANIMATION		REALISTIC MAGNETIC FIELD
BACKEND	3JS CODE	GESTURE CONTROLS	TRANSITION TO A-FRAME
	SCENE IMPORT	SCENE UPDATE	MULTIPLE SCENES UPLOAD
	CODE FOR ANIMATION	TESTING DIFFERENT FRAMEWORK	CONTROL WITH BUTTONS

DRAFT PRESENTATION

QR Code

Image Target

END OF PRESENTATION

THANK YOU

ATLAS-GTU TAI Agreement Workshop