

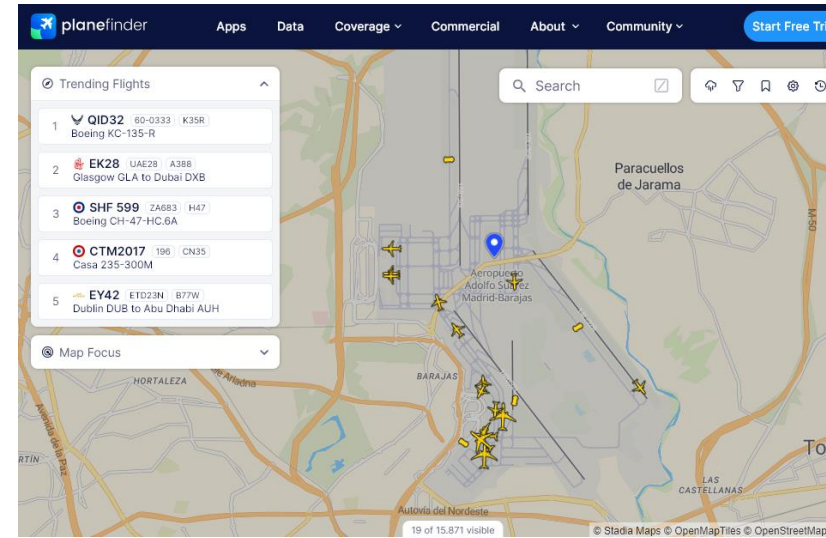
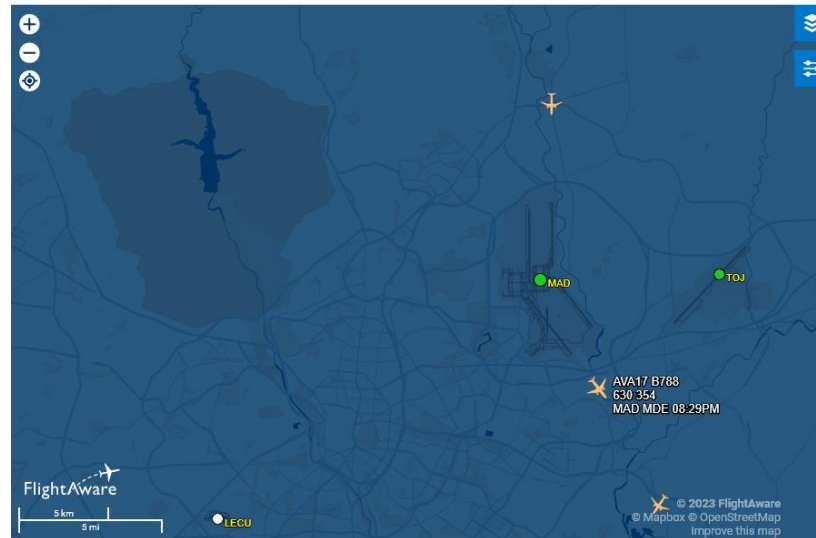
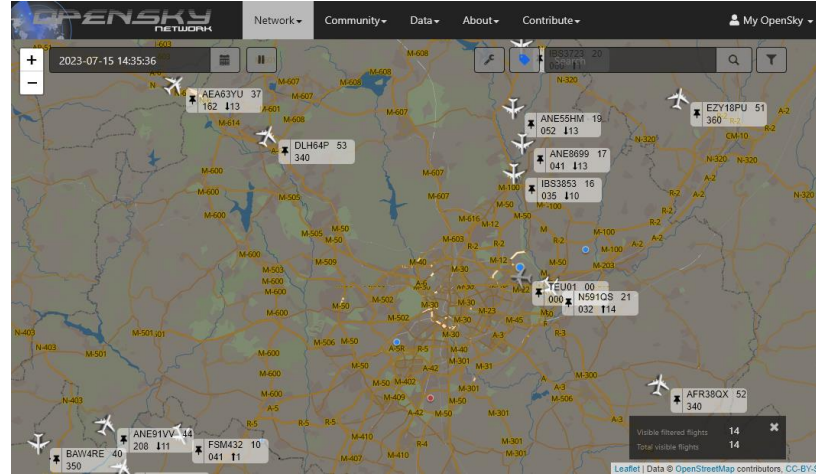
# VISUALIZACIÓN EN REALIDAD VIRTUAL DE DATOS AERONÁUTICOS CON CONTEXTO GEOESPACIAL

**TRABAJO FIN  
DE GRADO**

Autor : Víctor Jesús Temprano Hernández

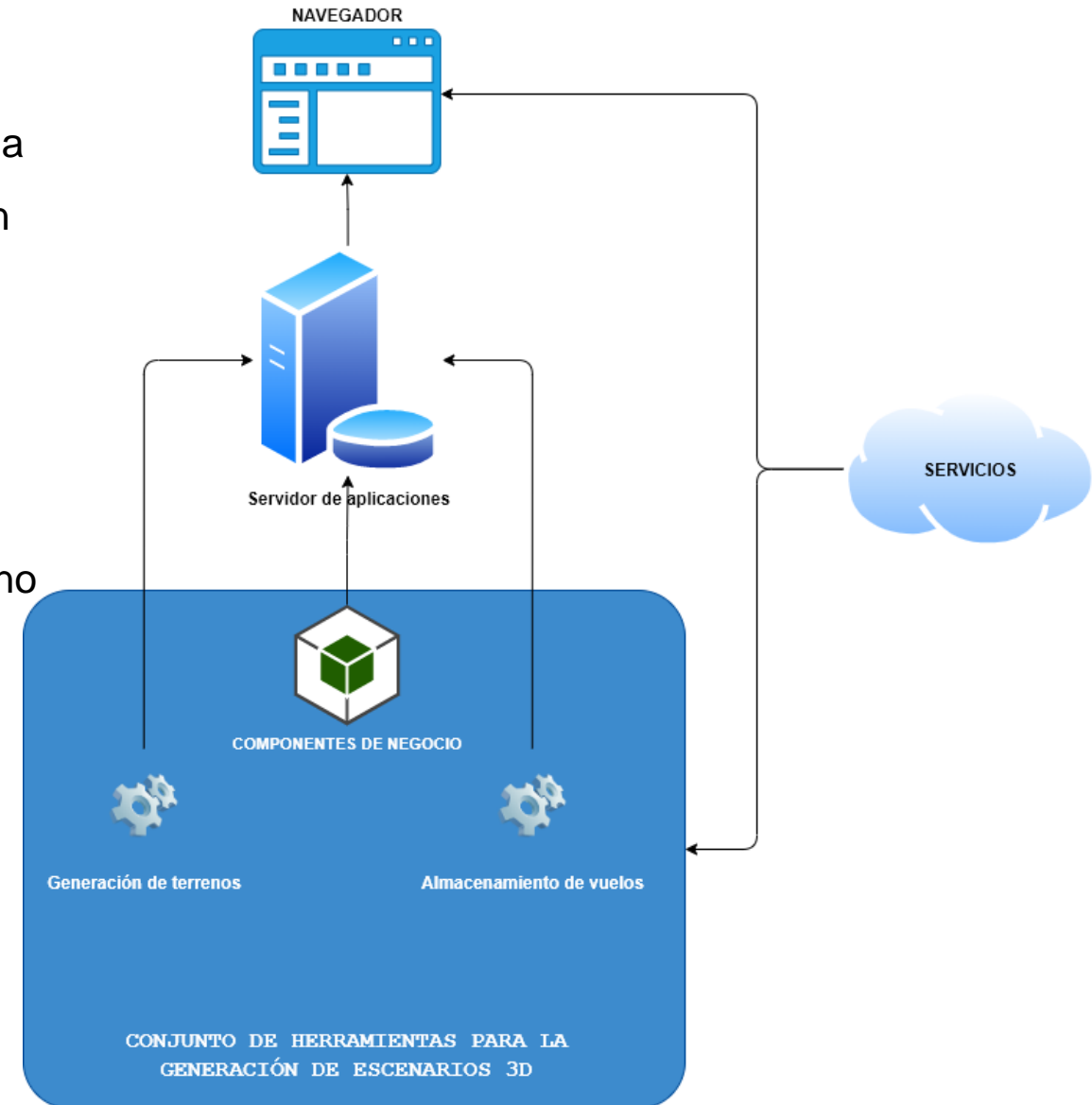
Tutor : Dr. Jesús M. González Barahona

# Contexto

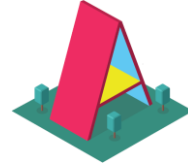


# Objetivos

- Creación de un conjunto de herramientas para facilitar la creación de escenarios 3D destinados a la visualización de información aérea con contexto geoespacial.
  - Utilizar A-Frame como tecnología principal
  - Posicionar vuelos en el escenario y animarlos
  - Crear terrenos realistas
  - Posicionar entidades geoespaciales sobre el terreno
  - Crear una interfaz gráfica para la visualización
  - Reproducir datos almacenados



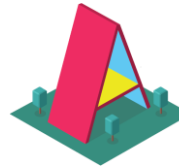
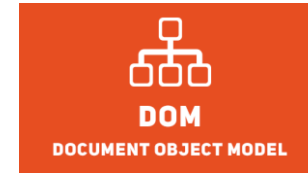
# A-Frame y Three.js



- Utiliza el patrón entidad componente, donde las entidades representan los objetos en la escena y, los componentes son bloques de funcionalidad que se agregan a las entidades para definir su comportamiento, apariencia o interacción
- Utiliza las etiquetas personalizadas de HTML5 para inicializar entidades y componentes de manera sencilla, sin la necesidad de escribir código JavaScript
- A través de estas etiquetas personalizadas de HTML, A-Frame encapsula y simplifica el uso de Three.js
- Permite mediante el uso de componentes alterar el comportamiento de las entidades

# Tecnologías relacionadas

**ADS-B**

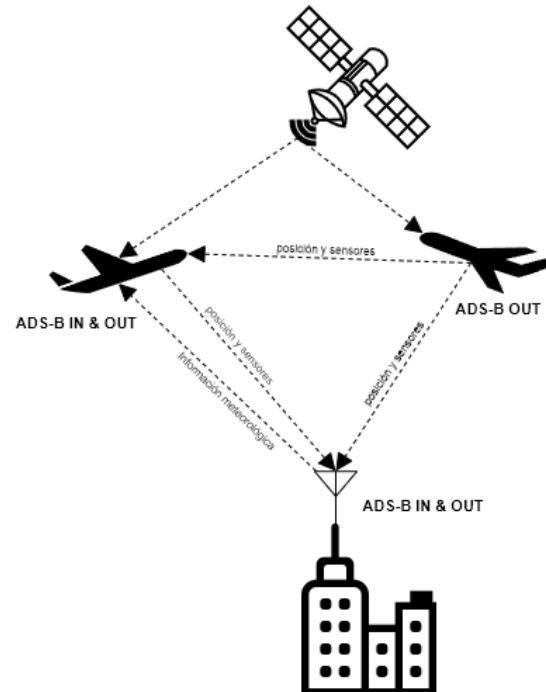


L<sup>A</sup>T<sub>E</sub>X



# ADS-B (Automatic Dependent Surveillance Broadcast)

- Transmite de manera automática periódicamente
- Depende del transpondedor y de los sensores
- Permite el seguimiento control de la aeronave
- Difusión de los datos a través de señales de radiofrecuencia en abierto y en todas direcciones
- Emite en la banda de los 1090Mhz
- Usan modulación digital
- Existen tres tipos de transpondedores:
  - IN
  - OUT
  - IN & OUT



Índice	Propiedad	Tipo	Descripción
0	icao24	string	Dirección del transpondedor en hexadecimal.
1	callsign	string	Nombre del avión o vuelo de 8 caracteres.
2	origin_country	string	País de origen.
3	time_position	int	Marca de tiempo UNIX en segundos de la última actualización.
4	last_contact	int	Marca de tiempo UNIX en segundos de la última trama ADS-B recibida.
5	longitude	float	Coordenada longitud WGS-84.
6	latitude	float	WGS-84 Coordenada latitud WGS-84.
7	baro_altitude	float	Altitud barométrica en metros.
8	on_ground	boolean	Indica si la posición se recuperó de un informe de superficie.
9	velocity	float	Velocidad en metros por segundo.
10	true_track	float	Orientación del avión en grados respecto al norte
11	vertical_rate	float	Velocidad vertical en metros por segundo
12	sensors	int[]	Identificadores de los receptores ADS-B que han contribuido.
13	geo_altitude	float	Altitud geométrica en metros.
14	squawk	string	Código del transpondedor.
15	spi	boolean	indicador para vuelos de propósito especial.
16	position_source	int	Fuente del vector de estado.
17	category	int	Enumerado de categoría del avión.

# HTML5

```
<head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1" />
  <title>Madrid Demo</title>
  <link rel="stylesheet" type="text/css" href="css/aframe.css">
  <script src="https://aframe.io/releases/1.4.0/aframe.min.js"></script>
  <script src="https://cdn.jsdelivr.net/gh/c-frame/aframe-extras@7.0.0/dist/aframe-extras.min.js"></script>
  <script src="https://unpkg.com/aframe-terrain-model-component@1.0.1/dist/aframe-terrain-model-component.min.js"></script>
  <script src="https://unpkg.com/aframe-look-at-component@0.8.0/dist/aframe-look-at-component.min.js"></script>
  <script src="https://unpkg.com/leaflet@1.9.2/dist/leaflet.js"></script>
  <script src="js/configuration/configurationModel.js" type="module"></script>
  <script src="js/configuration/madridconf.js" type="module"></script>
  <script src="js/gui/hud.js" type="module"></script>
  <script src="js/gui/custom-draggable.js" type="module"></script>
  <script src="js/gui/hover-scale.js" type="module"></script>
  <script src="js/map-ground/building-geometry.js" type="module"></script>
  <script src="js/mainscene.js" type="module"></script>
  <script src="js/map-ground/camera-height.js" type="module"></script>
  <script src="js/gui/oculus-test.js" type="module"></script>
  <script src="js/gui/tooltip-info.js" type="module"></script>
  <script src="js/gui/toolbar3d.js" type="module"></script>
  <script src="js/gui/track.js"></script>
  <script src="js/gui/camrender.js"></script>
  <script src="js/gui/canvas-updater.js"></script>
</head>
```



# HTML5

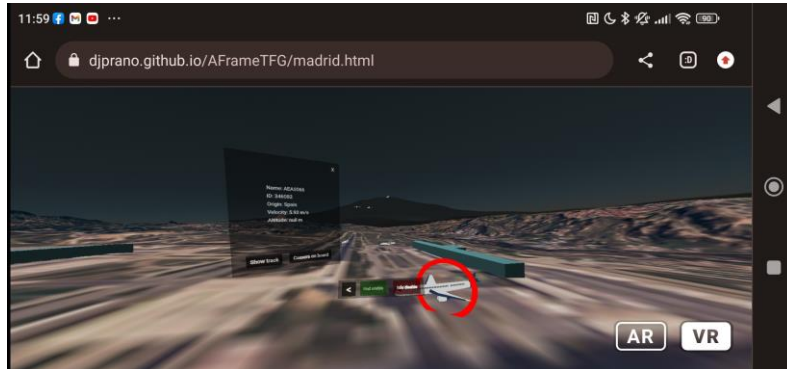
```

33 <body>
34   <a-scene main-scene>
35     <!-- Camera -->
36     <a-entity id="rig" position="0 0 0" movement-controls terrain-height>
37       <a-entity id="camera" hud camera look-controls="reverseMouseDown:false" cursor="rayOrigin: mouse; fuse: false"
38         raycaster="far: 2000; objects: .clickable" position="0 0 0" toolbar3d>
39     </a-entity>
40     <a-entity id="left-hand" oculus-touch-controls="hand: left" laser-controls="hand: left"
41       raycaster="far: 2000; objects: .clickable"></a-entity>
42     <a-entity id="right-hand" oculus-touch-controls="hand: right" laser-controls="hand: right"
43       raycaster="far: 2000; objects: .clickable"></a-entity>
44     <a-entity id="cameraOnBoarEntity" camera="active: false" camrender="cid:cameraOnBoard;fps:25" position="0 0 0"
45       rotation="0 -180 0"></a-entity>
46   </a-entity>
47   <a-sky id="sky" src="#skyImage" theta-length="90" radius="1000"></a-sky>
48   <a-sphere id="moon" material="shader: flat; color: #fef7ec" radius="10" position="-96 350 -238"
49     light="type: directional; color: #fef7ec; intensity: 0.65"></a-sphere>
50 </a-scene>
51 </body>
52 </html>

```



# Resultados

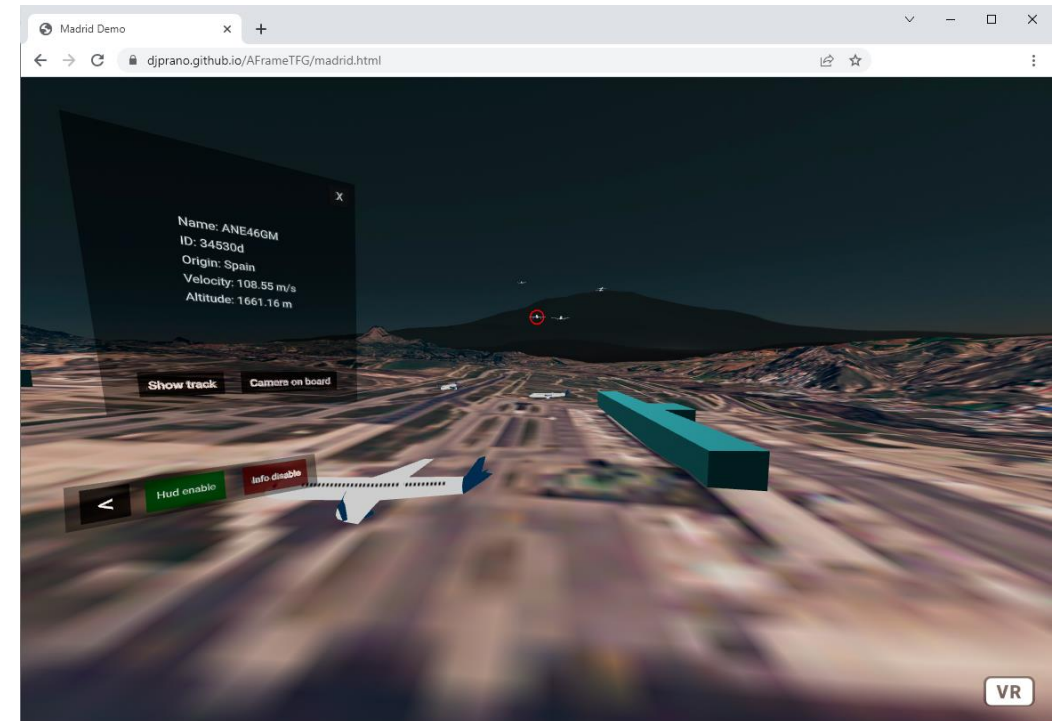


Móvil



Gafas Oculus

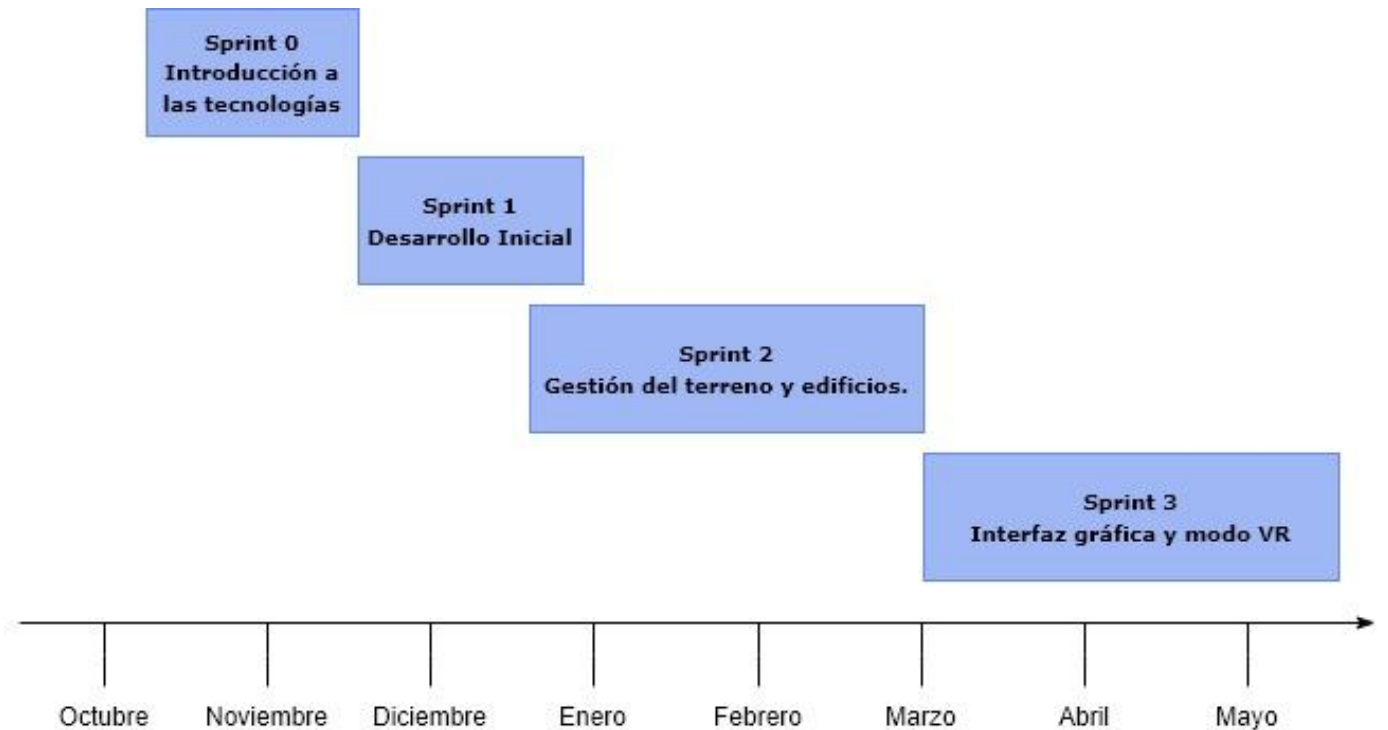
Escritorio



# Proceso de desarrollo

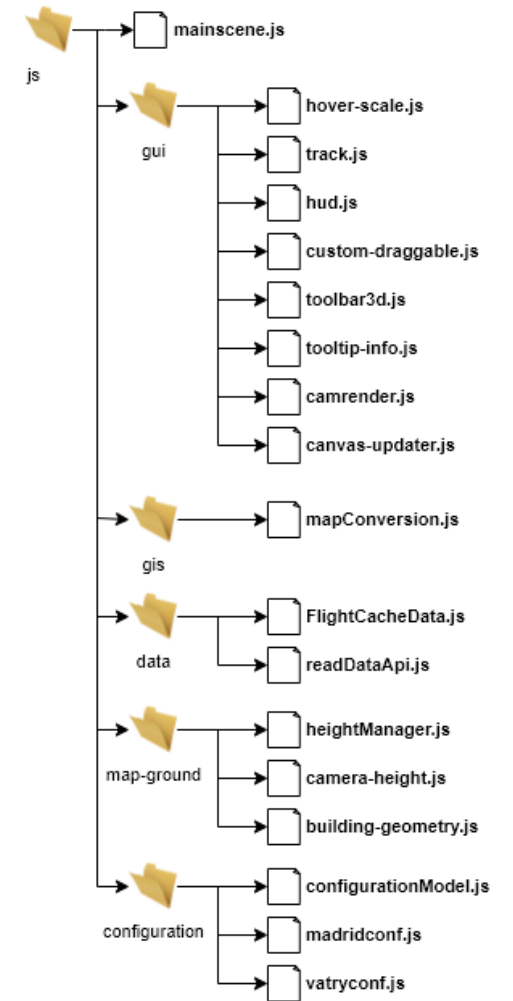
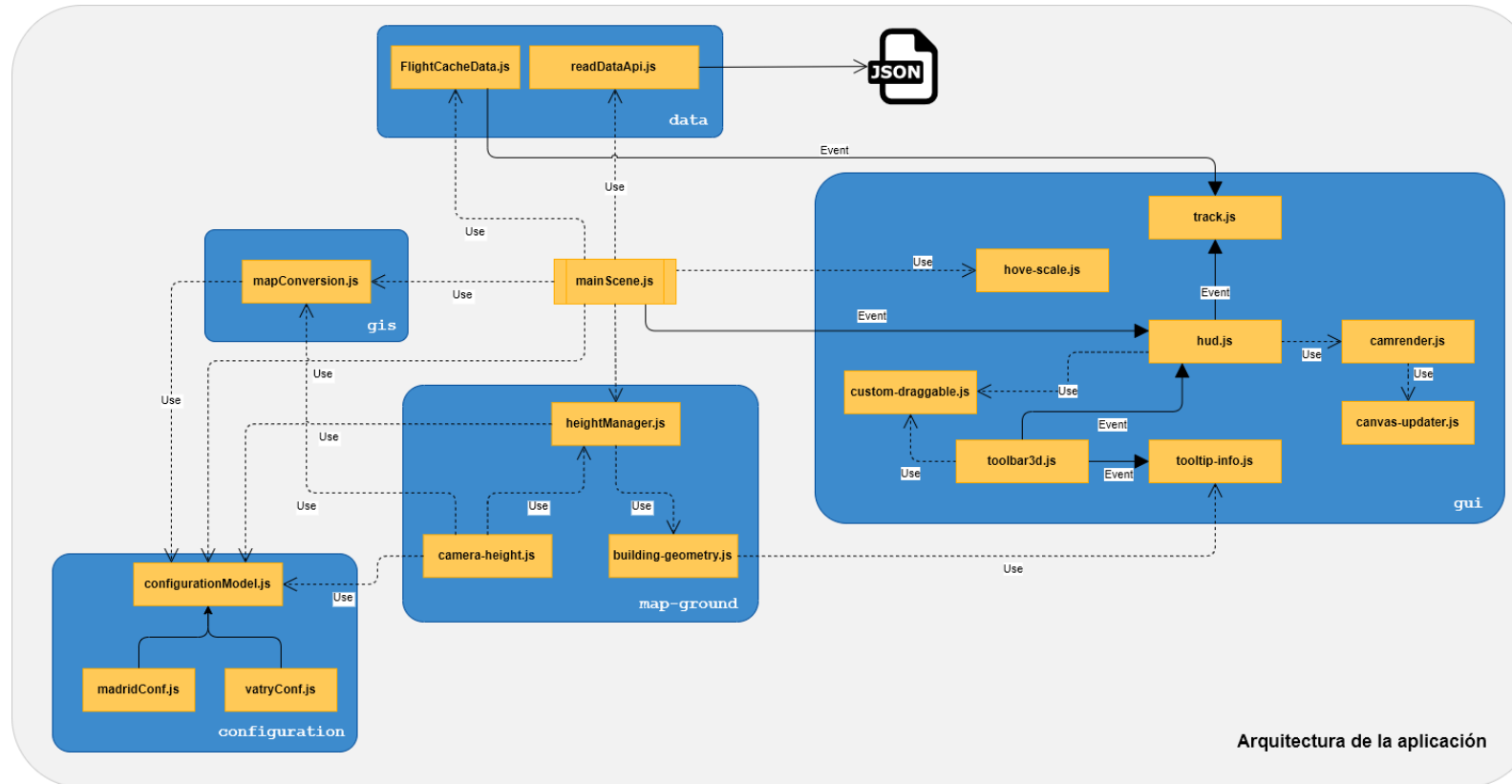
## Metodología ágil

- Inspirado en Scrum
- Se definen tres roles principales:
  - Product Owner (Tutor)
  - Scrum Master (Tutor)
  - Developer (Autor)
- Integración continua a través de Github Pages
- Sprint Review y Sprint Planning
- Reuniones intermedias.



# Diseño e implementación

## Arquitectura general





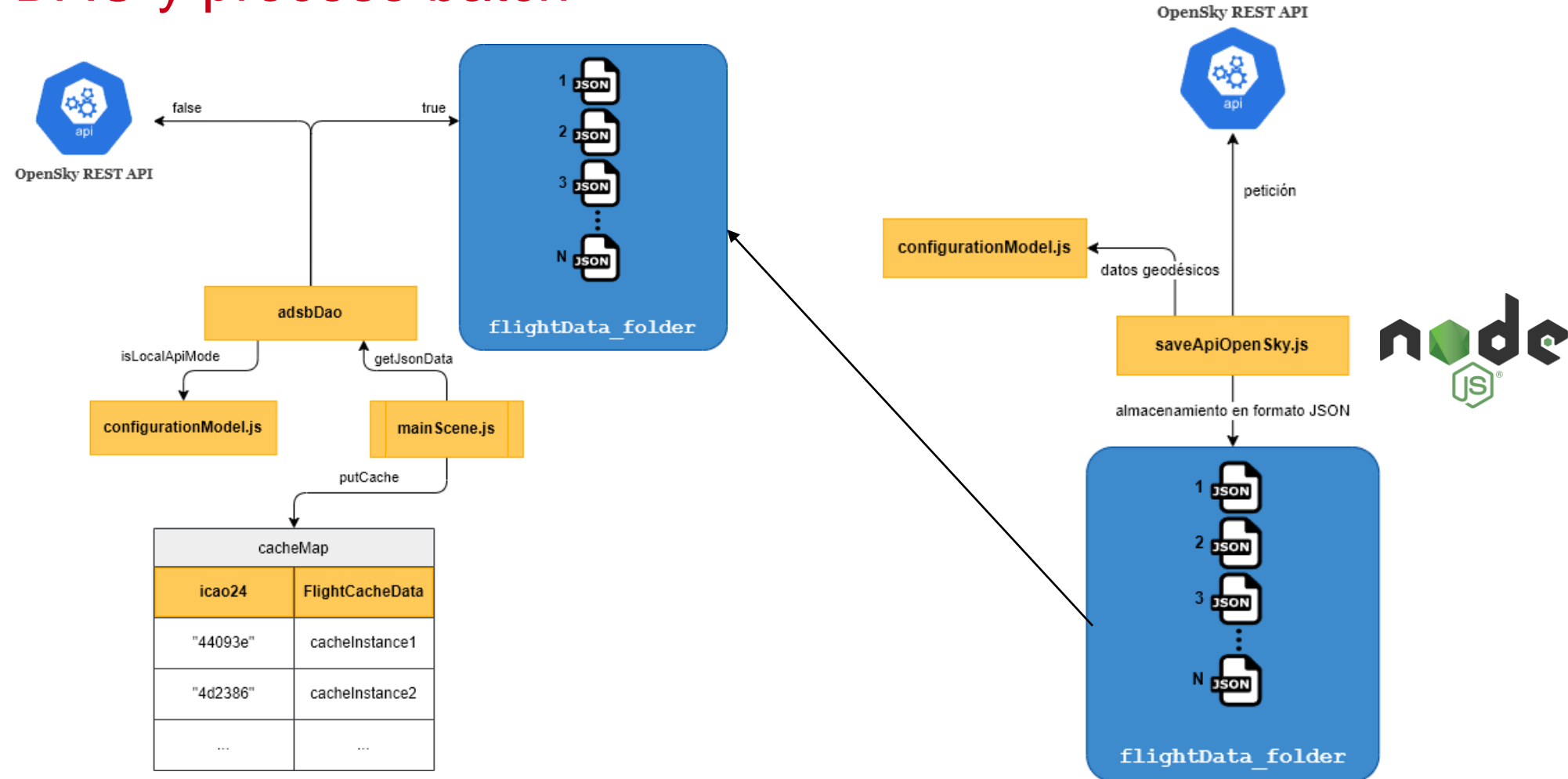
# Gestor principal de la escena

## Mainscene

```
heightManager.createMapGround();  
// Establecemos la función de intervalo cuando se carga el terreno y  
// la inicializamos a vacío para no provocar condiciones de carrera.  
this.throttledFunction = ()=>{  
  //Nothing  
};  
// Cuando se cargue el terreno establecemos la correcta.  
heightManager.addTerrainLoadedListener(() => {  
  this.throttledFunction = AFRAME.utils.throttle(this.invertalEvent, intervalTime, this);  
});  
  
rig.setAttribute('position', initCamPosition);  
},  
  
invertalEvent: function () {  
  // Called every second.  
  adsbDao.getJsonData().then(jsonData => updateData(jsonData))  
},  
  
tick: function (t, dt) {  
  this.throttledFunction();  
},
```

# Datos de vuelos

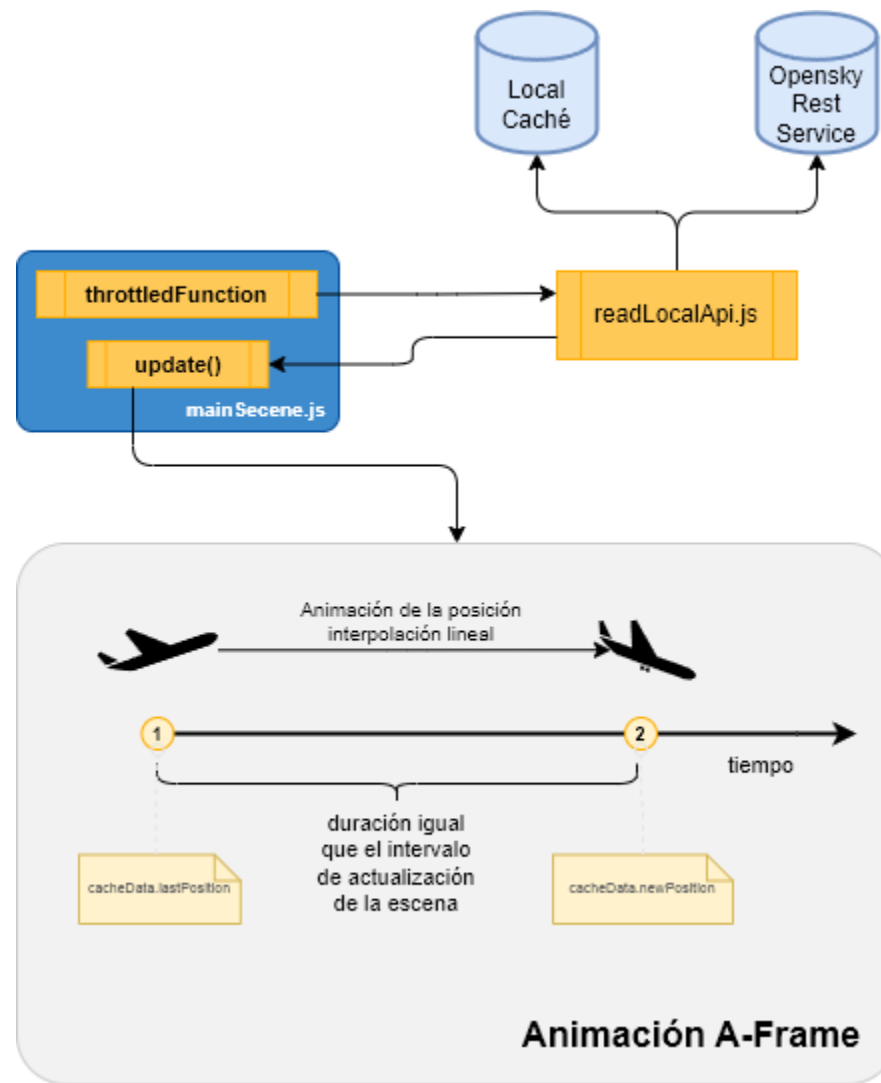
## DAO y proceso batch



# Movimiento fluido de aviones

## Mainscene

```
if (cacheData.lastPosition !== cacheData.newPosition) {
  entityEl.setAttribute('animation__000', {
    property: 'position',
    from: cacheData.lastPosition,
    to: cacheData.newPosition,
    autoplay: true,
    loop: 0,
    easing: 'linear',
    dur: intervalTime
  });
}
```



# Gestión de la configuración

## Parametrización de la aplicación

```
<script src="js/configuration/configurationModel.js" type="module"></script>
<script src="js/configuration/madridconf.js" type="module"></script>
```

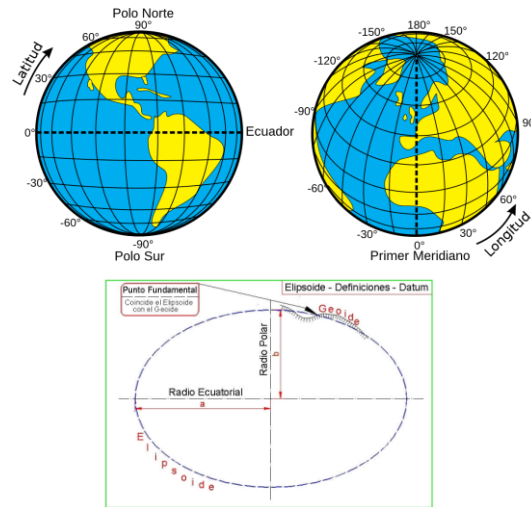
```
import * as configuration from './configurationModel.js';
//Establece las coordenadas geodésicas del escenario (latmin, latmax, longmin, longmax).
configuration.setMerConfig(40.0234170, 40.7441446, -4.2041338, -3.2538165);
configuration.setCamPosition(40.50, -3.54); //posición de la cámara.
configuration.setBuildingFileName('madrid_building'); //carpeta de edificios.
configuration.setFlightLocalFolder('_madrid'); //carpeta caché de vuelos.
configuration.setMapRaster('Madrid_raster.jpg'); //fichero raster del terreno
configuration.setMapDem('madrid_dem.bin'); //fichero de alturas del terreno
configuration.setLocalApiMode(true); //establece el modo offline datos cacheados.
configuration.setDaoInterval(1000); //intervalo de refresco.
configuration.setDaoLocalIndex(0); //índice del fichero por donde comienza.
configuration.setApiUser('xxxx'); //Establece el usuario para el modo online
configuration.setApiPassword('xxxxx'); //Establece la contraseña para el modo online
```



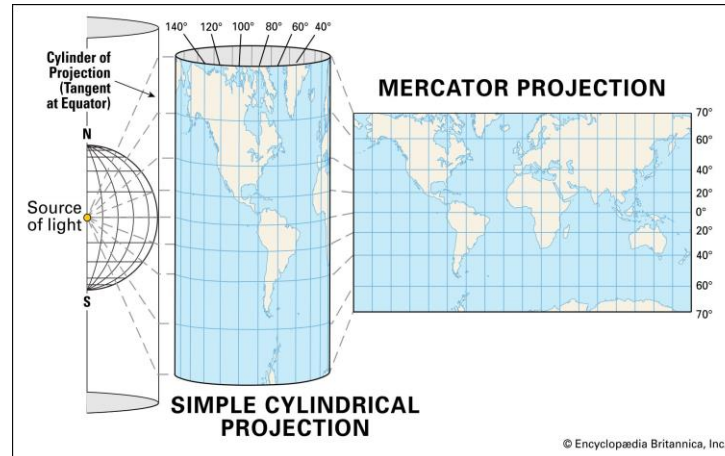
# Sistema de información geográfica

## GIS

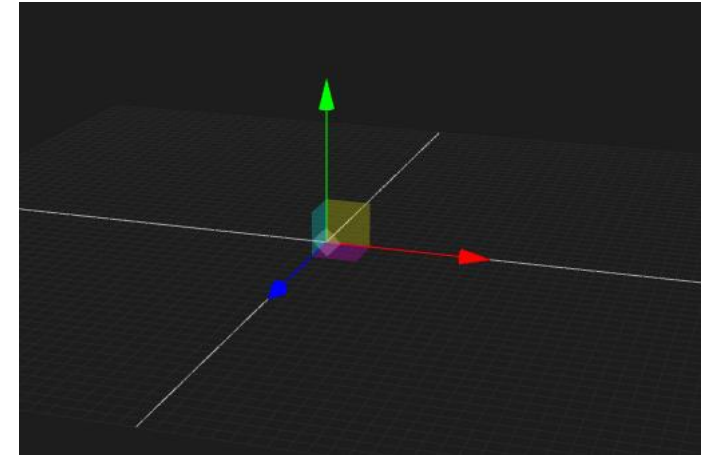
### Geodésicas (WGS84)



### Proyectadas



### Mundo 3D

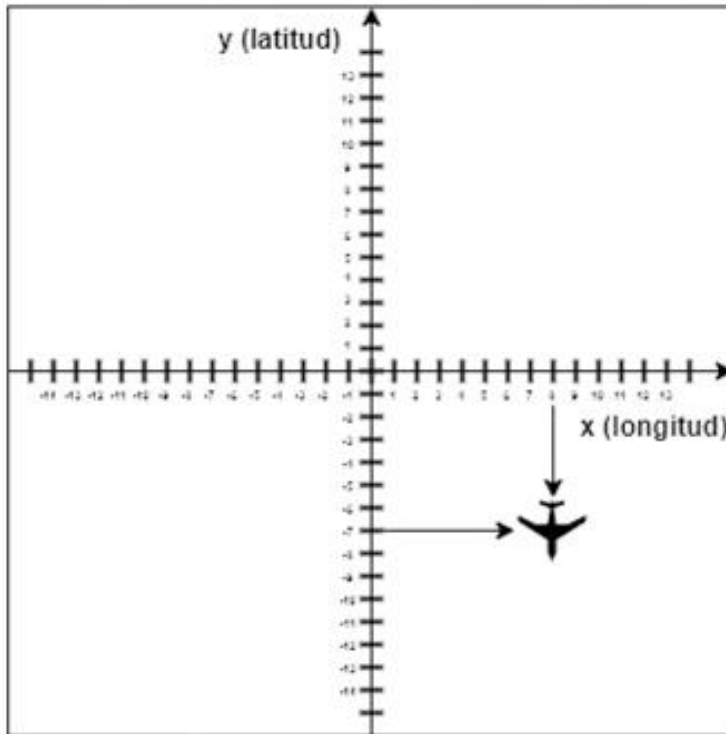


# Sistema de información geográfica

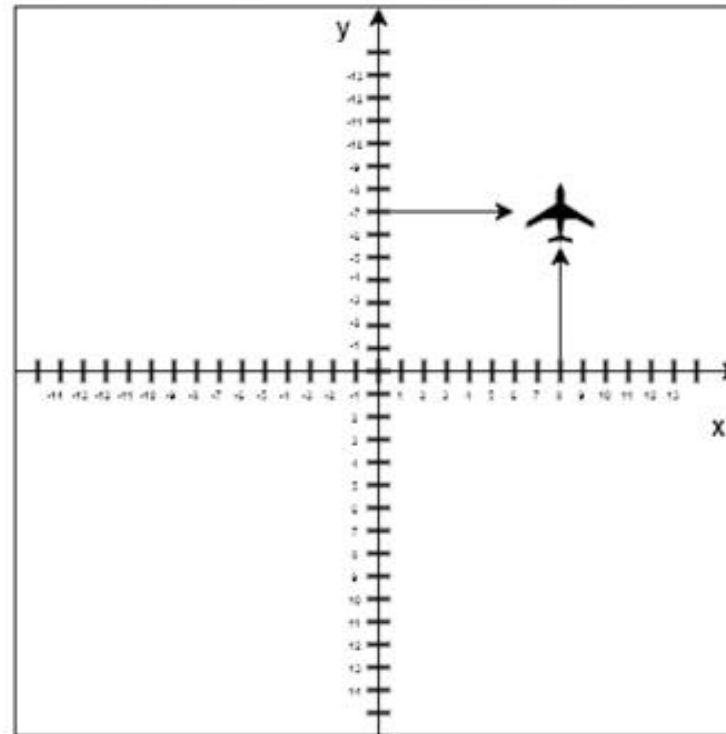
## GIS Problema 1

Asignar el eje y cartesiano al eje Z e invertirlo

Representación del suelo en coordenadas Geoespaciales



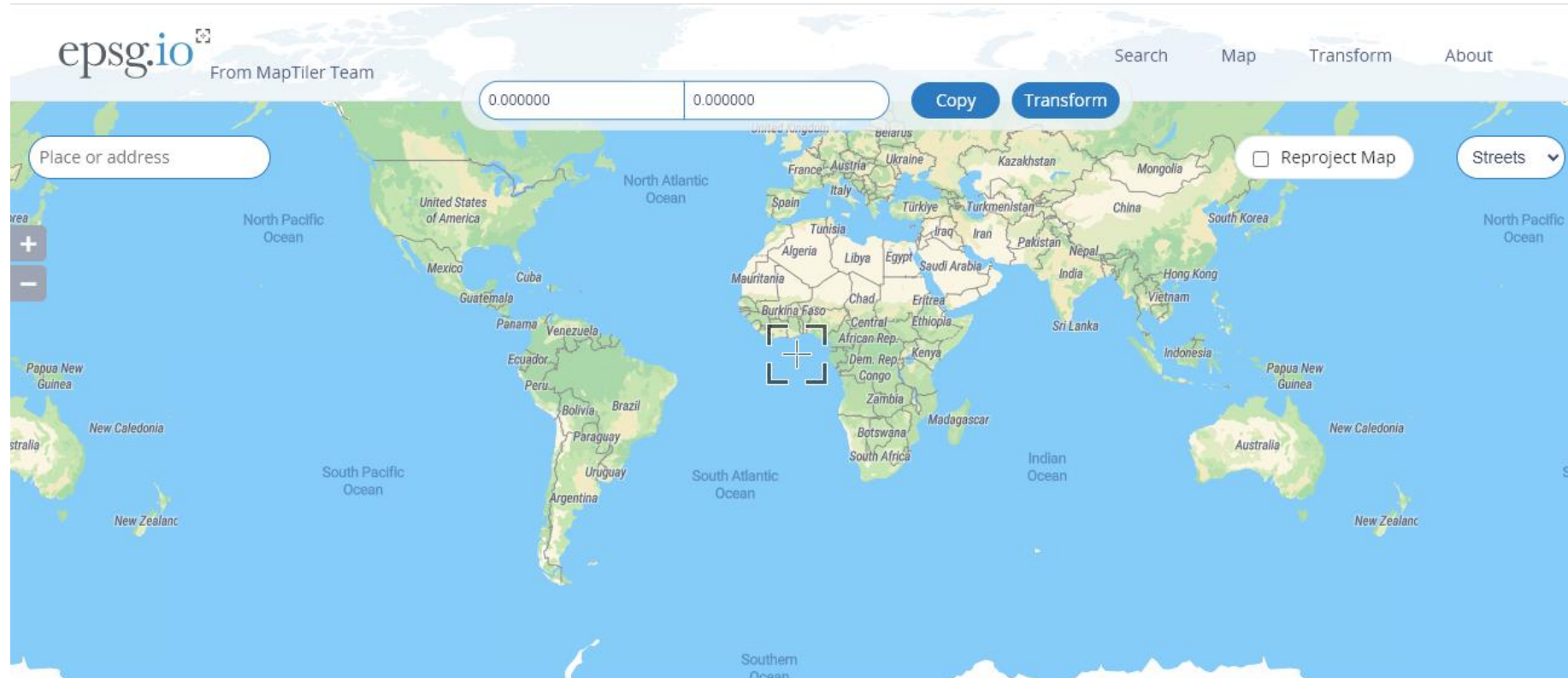
Representación del suelo en coordenadas A-Frame



# Sistema de información geográfica

## GIS Problema 2

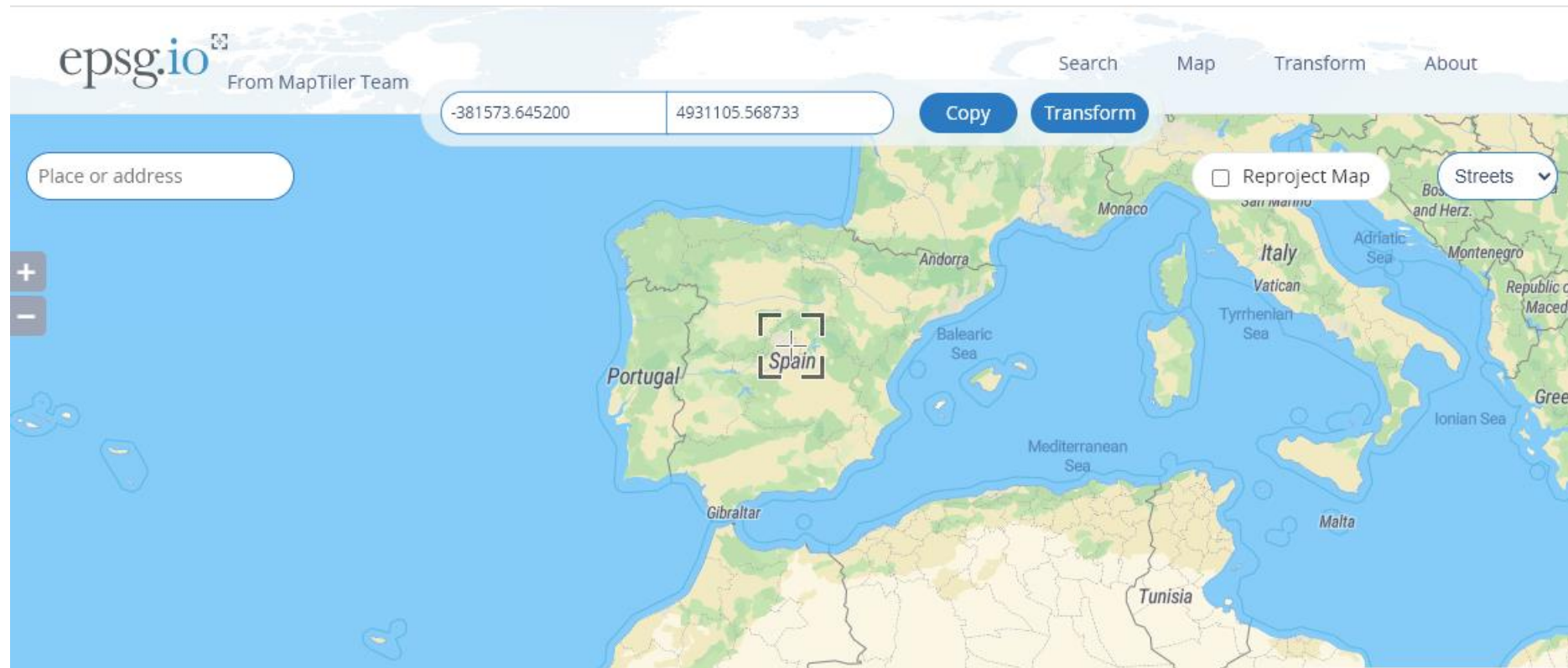
Coordenadas descentralizadas y dimensiones excesivas



# Sistema de información geográfica

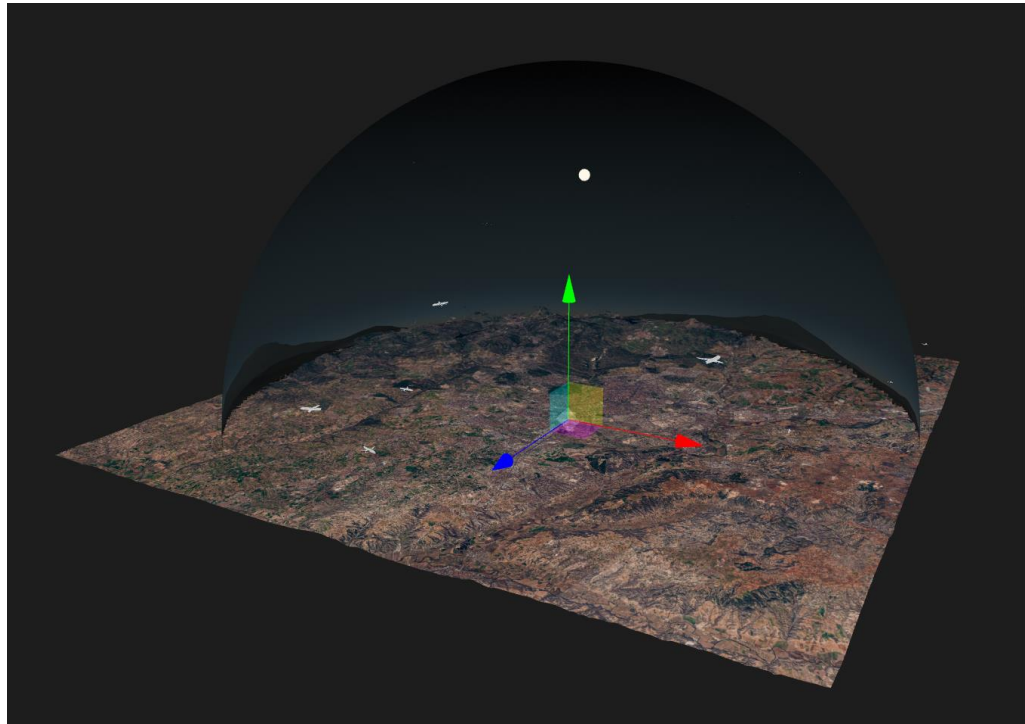
## GIS Problema 2

Debemos centrar el escenario y realizar operaciones afines.



# Sistema de información geográfica

## GIS solución



$$X_{3DWorld} = \frac{\text{mercatorVector}_x - \text{displacement}_x}{\text{FACTOR}}$$

$$Y_{3DWorld} = \frac{\text{mercatorVector}_z - \text{displacement}_y}{\text{FACTOR}}$$

$$3DWorld = \left[ X_{3DWorld}, \frac{\text{height}}{\text{FACTOR}}, -Y_{3DWorld} \right]$$

$$X_{\text{Mercator}} = (\text{world3DVector}_x \times \text{FACTOR}) + \text{displacement}_x$$

$$Y_{\text{Mercator}} = (\text{world3DVector}_z \times \text{FACTOR}) + \text{displacement}_y$$

$$\text{Mercator} = \left[ X_{\text{Mercator}}, \text{world3DVector}_y \times \text{FACTOR}, -Y_{\text{Mercator}} \right]$$

# Gestor del terreno

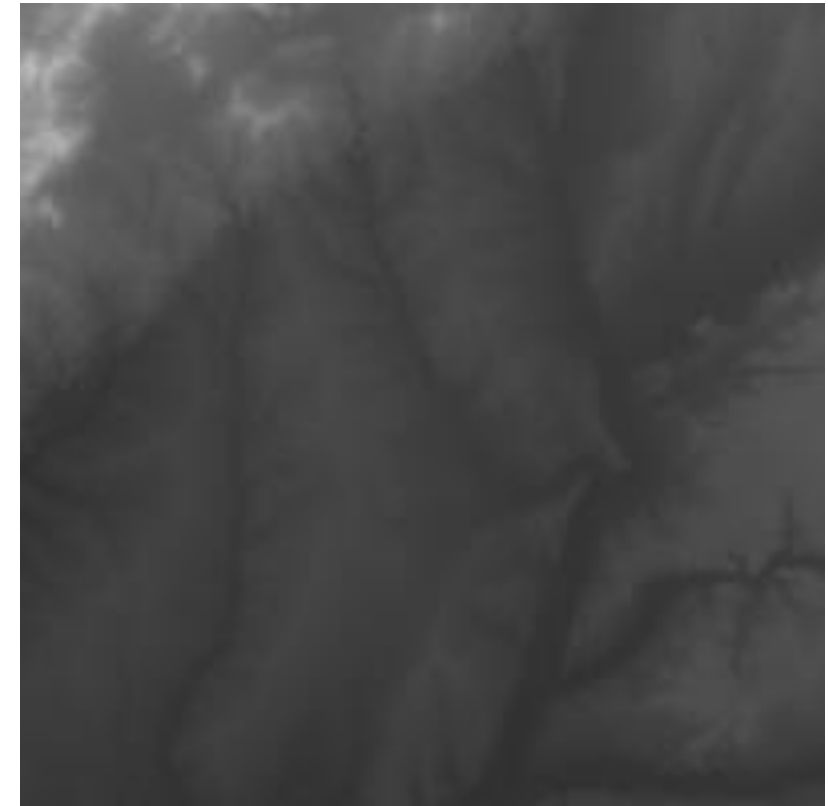
## Map-ground

- Es responsable de:
  - La generación del mallado del terreno de la escena.
  - La carga de la textura ráster del terreno de la escena.
  - La generación de los edificios sobre la superficie del terreno.
  - Obtener la matriz de alturas cargada y proporcionar una API al resto de módulos para el cálculo eficiente de altura en cualquier punto del escenario.
  - Hacemos uso del componente  
<https://github.com/bryik/aframe-terrain-model-component>



# Gestor del terreno

## COPERNICUS y GDAL





# Gestor del terreno

## Raster

```
var geometry = ee.Geometry.Rectangle(-4.204133836988655, 40.023417003380956,
  ↪ -3.253816454176155, 40.744144594569384);
// Convierte la geometría a un objeto Feature y establece un nombre
var rectangulo = ee.Feature(geometry, {nombre: 'Mi rectángulo'});
// Añade el rectángulo a la vista del Mapa
Map.addLayer(rectangulo, {}, 'Rectángulo');
var ColeccionSentinel = ee.ImageCollection('COPERNICUS/S2').filterDate('2019-01-01',
  ↪ '2019-02-28');
var Mosaico = ColeccionSentinel.mosaic();
Map.addLayer(Mosaico, {max: 5000.0, min: 0.0, gamma: 1.0, bands: ['B4', 'B3', 'B2']},
  'Composicion RGB');
// Crear una imagen RGB utilizando las bandas B4, B3 y B2
var RGB = Mosaico.visualize({bands: ['B4', 'B3', 'B2'], max: 5000, min: 0, gamma: 1.0});
// Crea un objeto Projection a partir de la identificación EPSG
var epsg3857 = 'EPSG:3857';
// Descargar la imagen RGB en formato GeoTIFF
Export.image.toDrive({image: RGB, description: 'Sentinel2_RGB', scale: 20, crs: epsg3857, region:
  ↪ geometry, maxPixels: 28710052848});
```



Google Earth Engine

# Gestor del terreno

## Raster



# Gestor del terreno

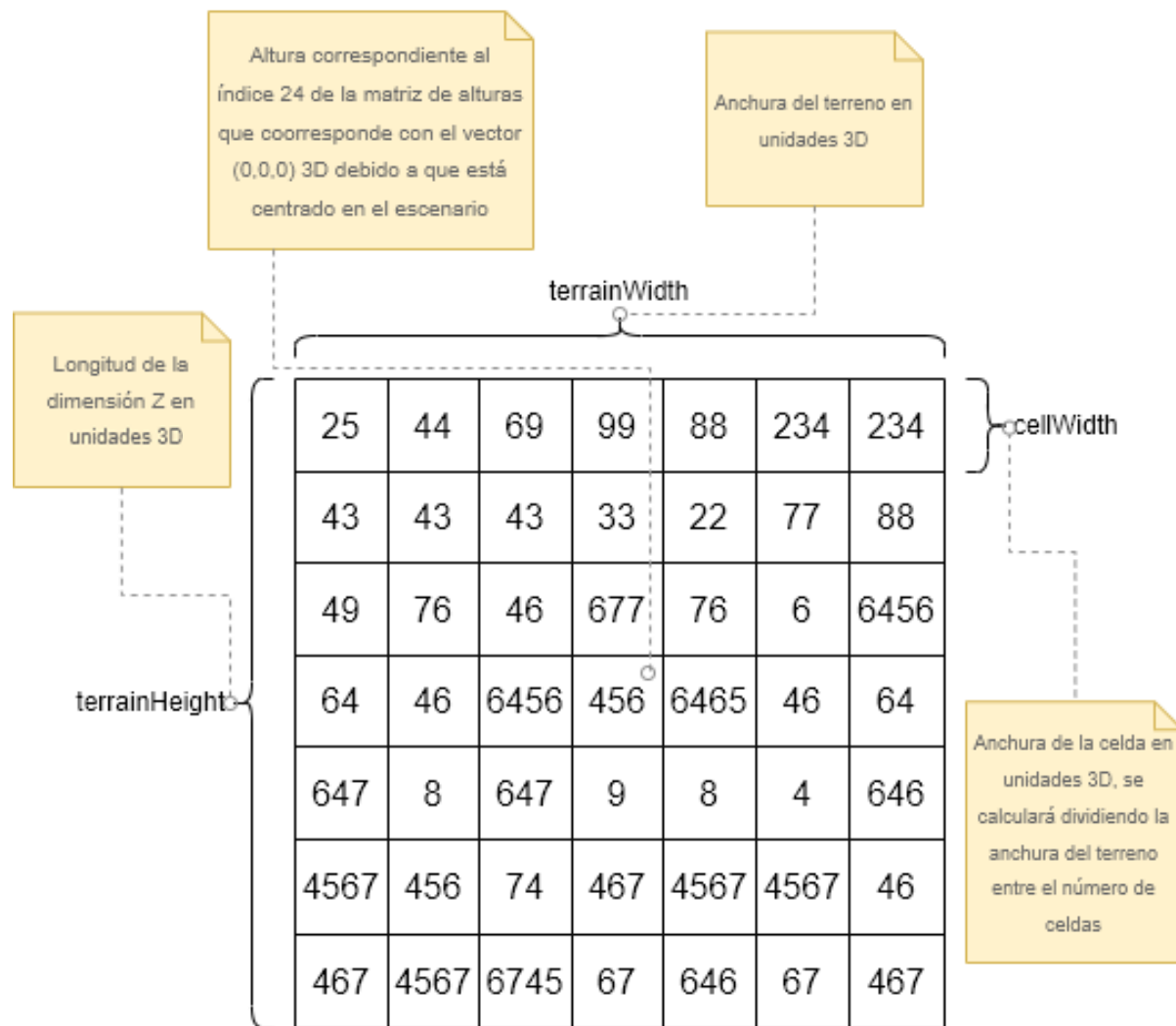
## Matriz de alturas

$$index_x = \text{round} \left( \frac{\text{vector3D}_x + \frac{\text{terrainWidth}_{3D}}{2}}{\text{cellWidth}} \right)$$

$$index_y = \text{round} \left( \frac{\text{vector3D}_z + \frac{\text{terrainHeight}_{3D}}{2}}{\text{cellHeight}} \right)$$

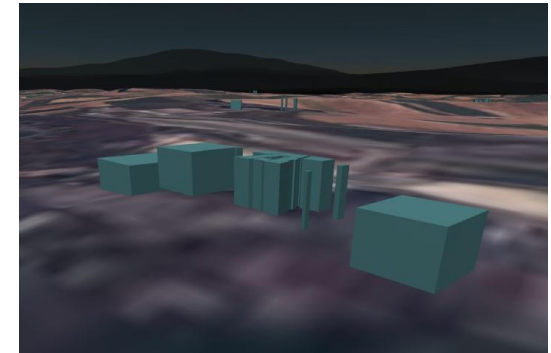
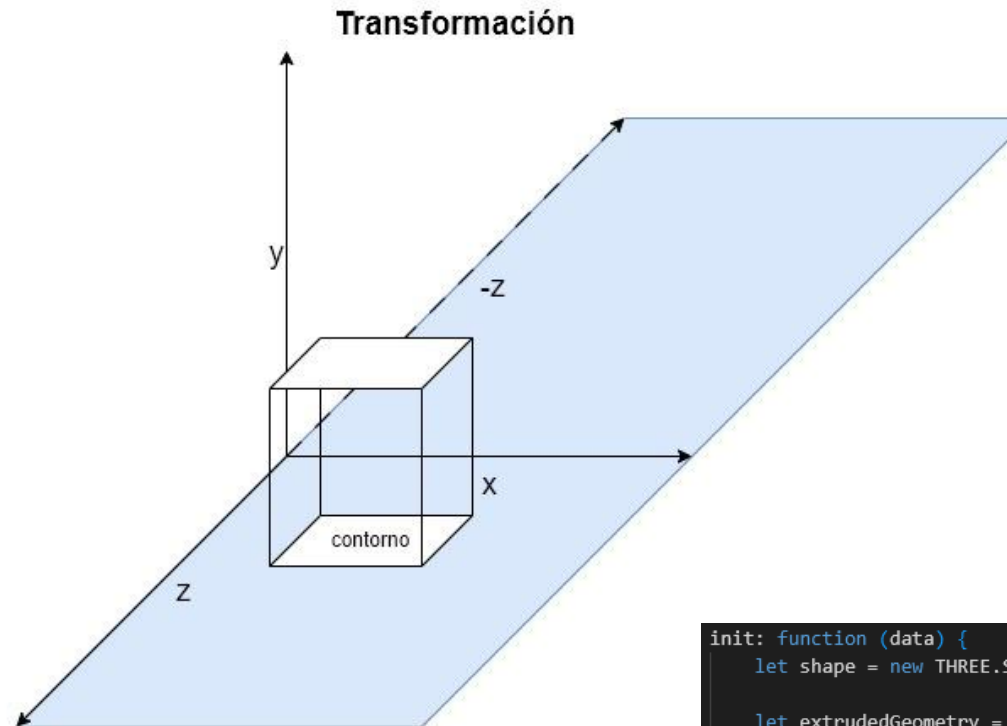
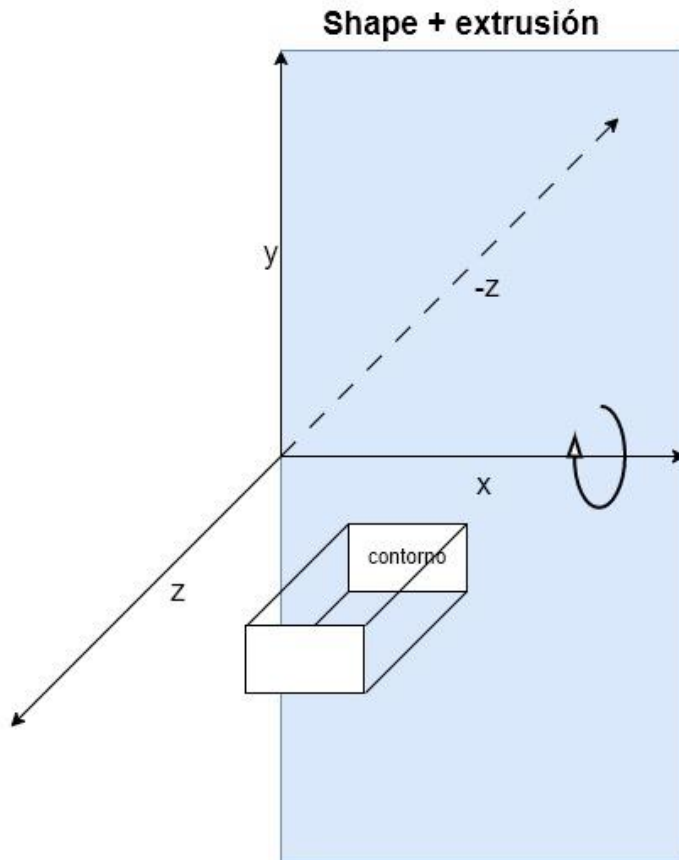
$$index = (index_y \times (\text{gridWidth} + 1)) + index_x$$

$$height = \frac{\text{magnification}_y \times \text{heightMatrix}[index]}{65535}$$



# GUI

## Geometría edificio



```
init: function (data) {
  let shape = new THREE.Shape(data.points);

  let extrudedGeometry = new THREE.ExtrudeGeometry(shape, {
    depth: data.height,
    bevelEnabled: false
  });
  extrudedGeometry.rotateX(Math.PI / 2);
  extrudedGeometry.translate(0, data.height + data.terrainHeight, 0);
  this.geometry = extrudedGeometry;
}
```

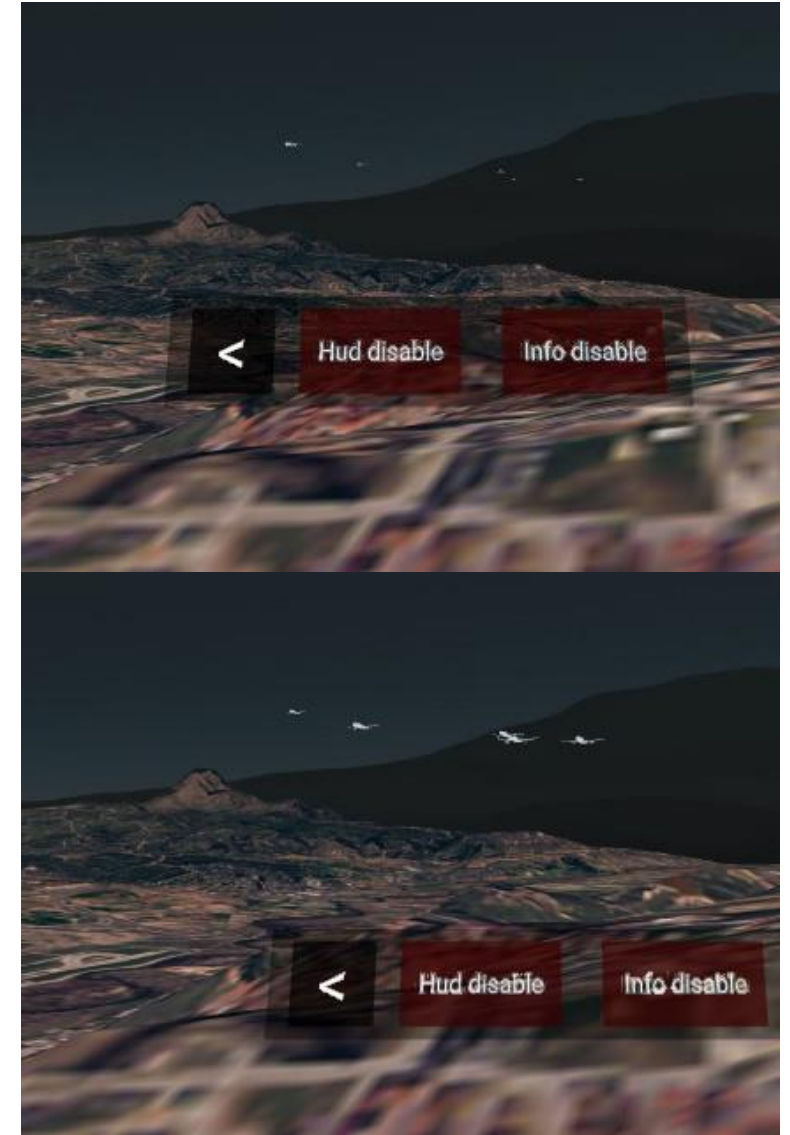


# GUI

## Amplificación de objetos

- Cuando las entidades superan una distancia con el usuario principal, comenzamos a escalar el modelo glTF de manera lineal.
- Sacrificamos realismo por funcionalidad.
- Una posible mejora es añadir la funcionalidad en la barra de herramientas para activar o desactivar este componente.

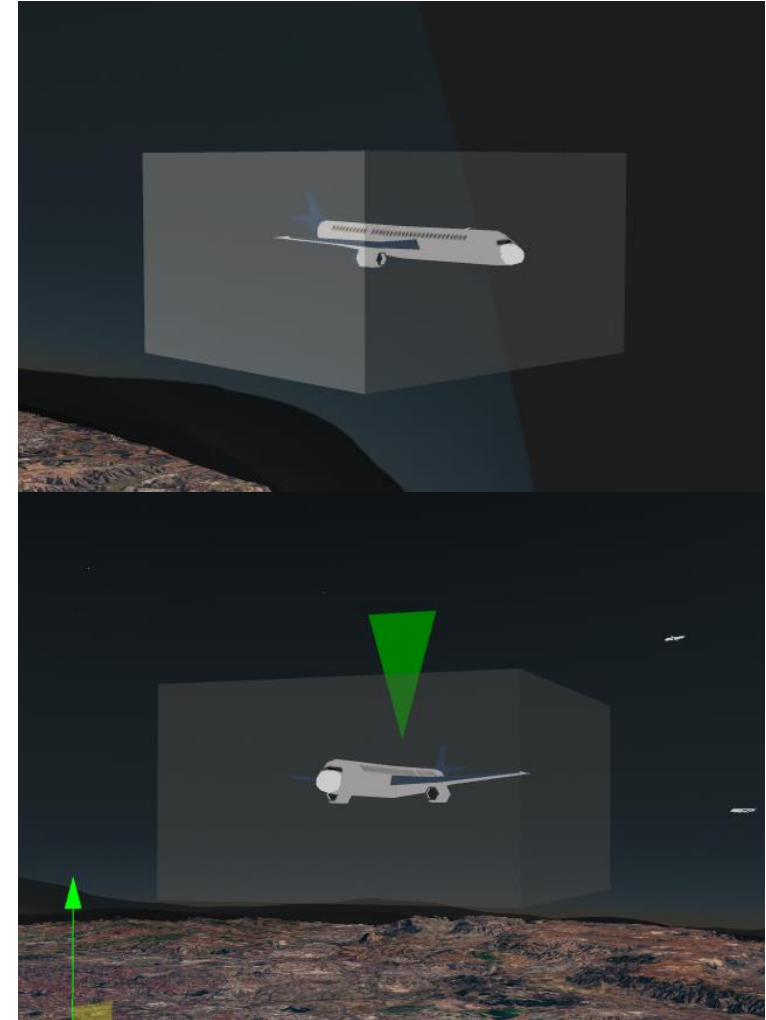
$$\text{Factor de Ampliación} = \left( \frac{\text{distancia} - \text{distancia umbral}}{\text{divisor ampliación}} \right) + 1$$



# GUI

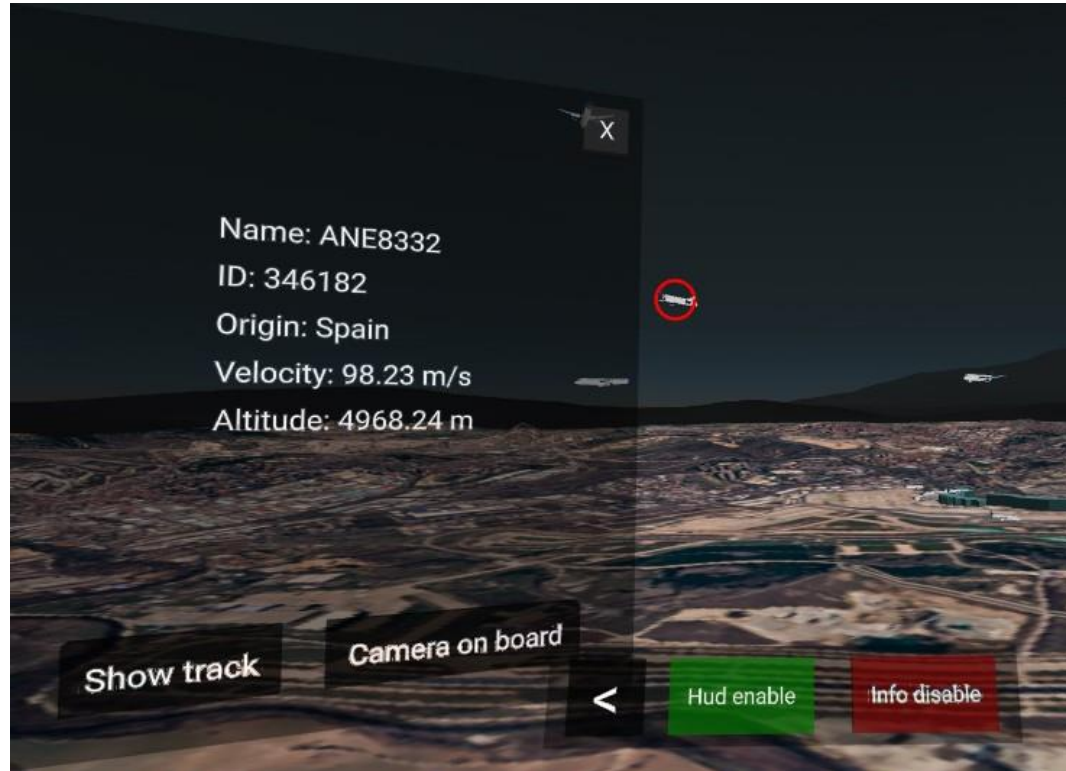
## Ayudas a la selección de vuelos

- Se crea un cubo envolvente sobre la entidad a través de Three.js que nos ofrece una API para crear cajas envolventes sobre la geometría 3D
- Creamos los gestores de eventos sobre estas cajas en vez de sobre el modelo glTF.
- Añadimos un indicador en forma de triángulo para indicar al usuario que tiene el selector encima de la entidad y puede seleccionarla.



# GUI

## HUD

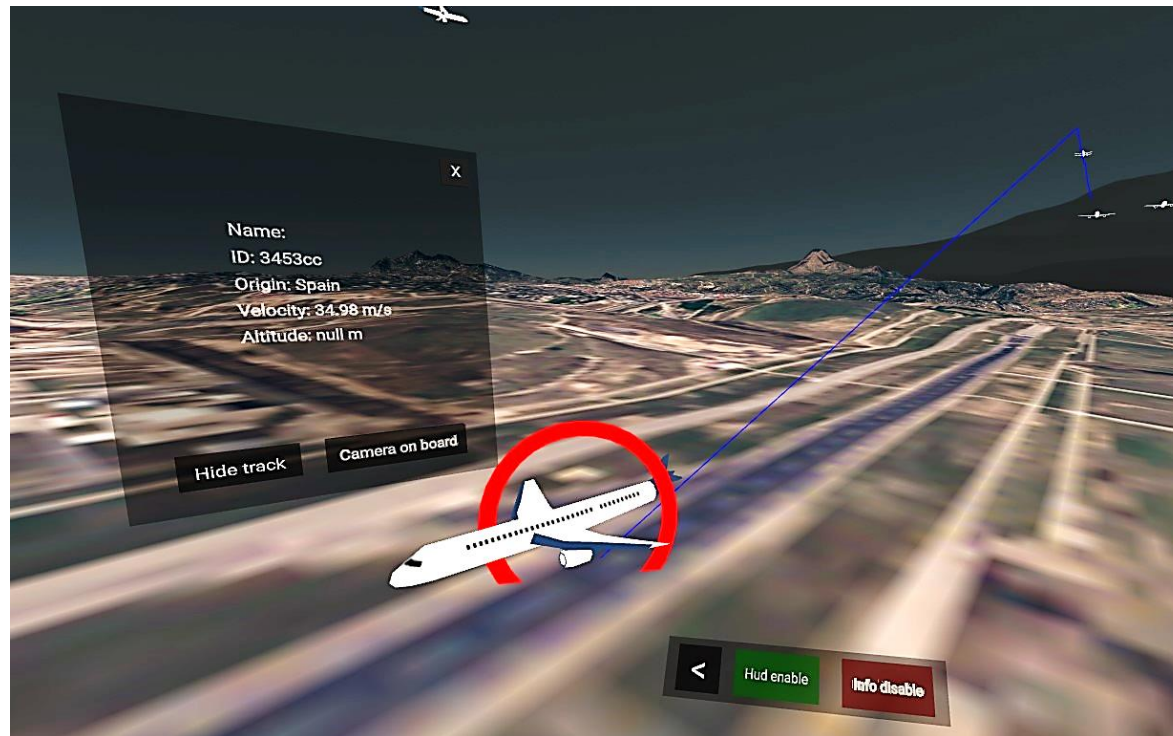




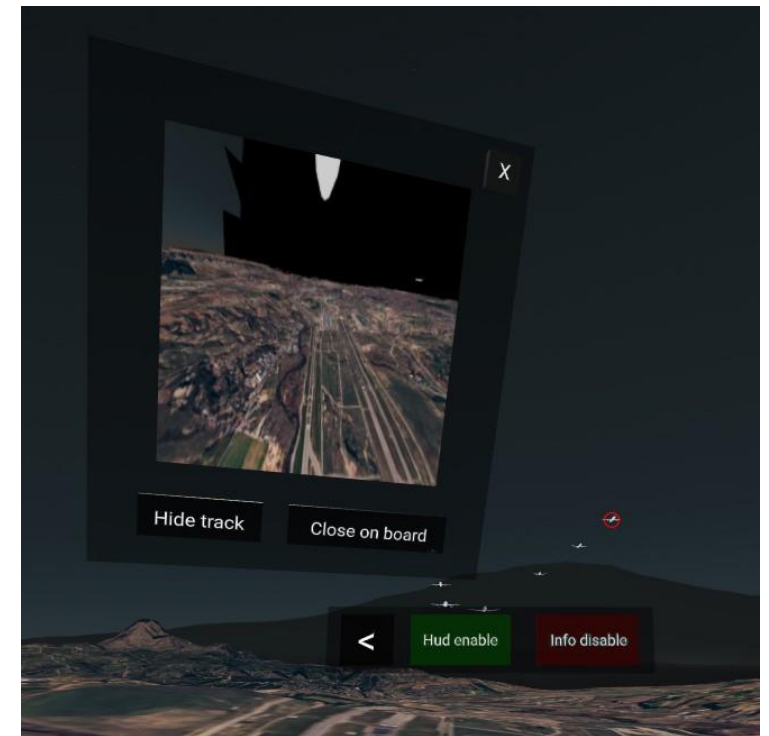
# GUI

## Trayecto y cámara de a bordo

### Funcionalidad trayecto

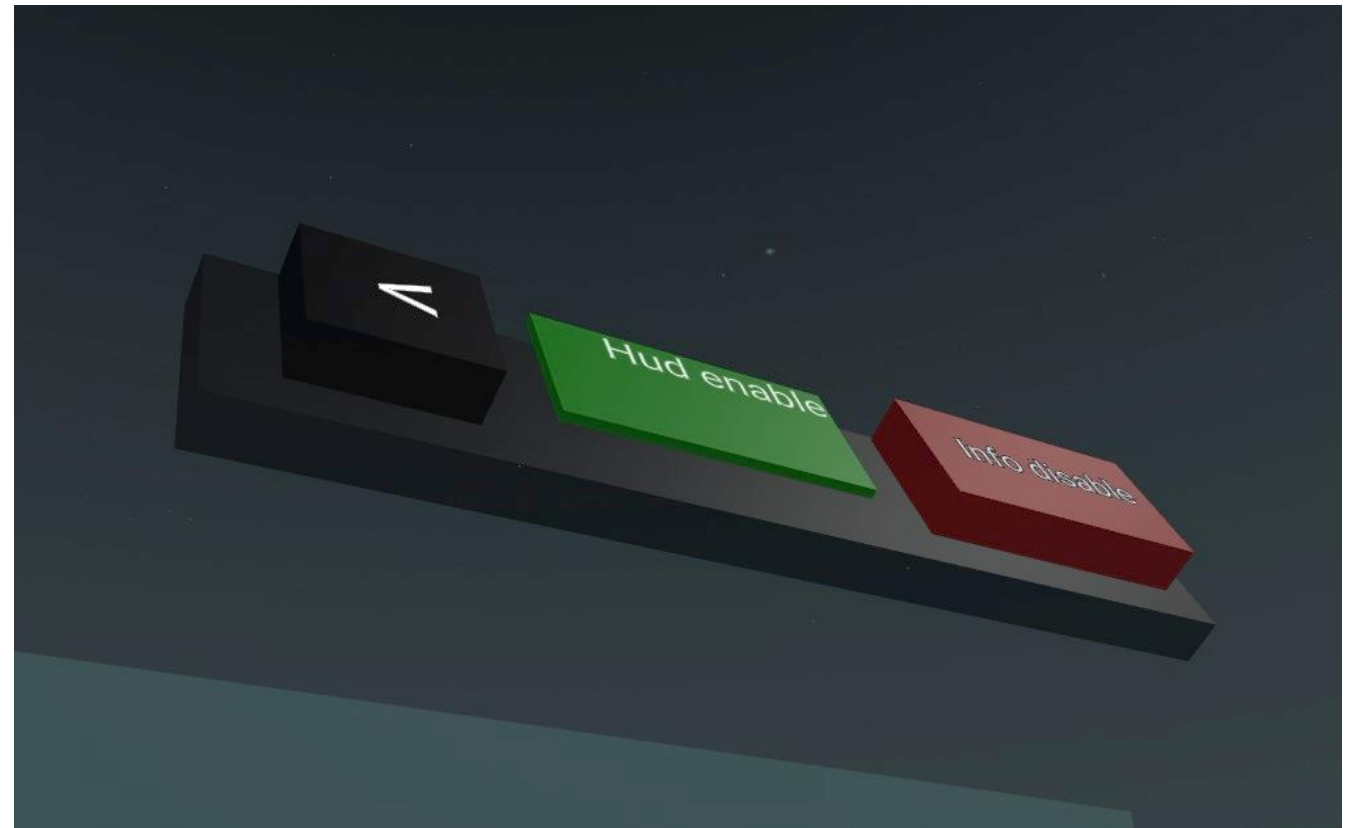
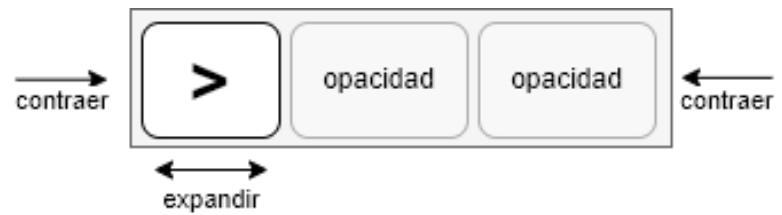
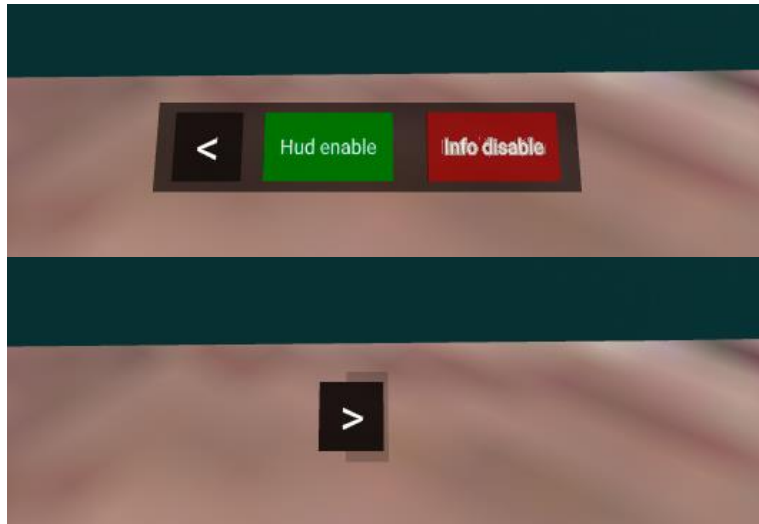


### Cámara de a bordo



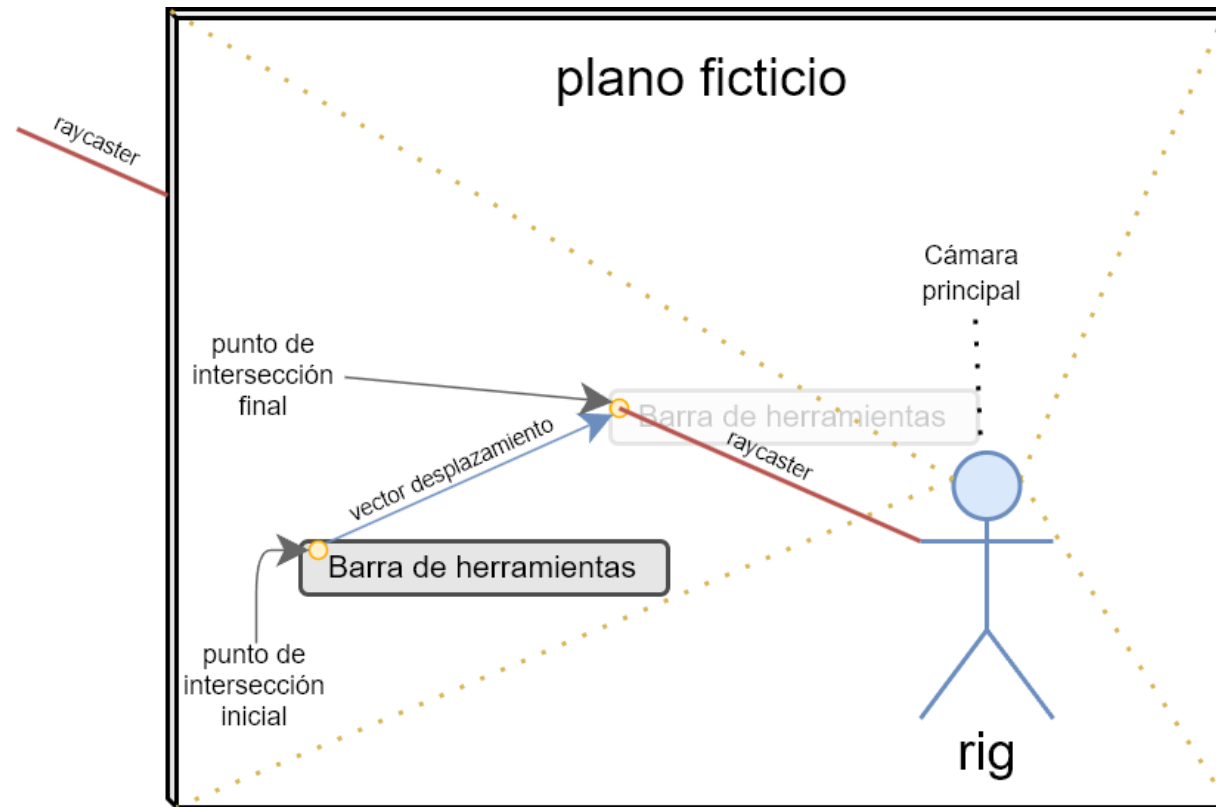
# GUI

## Barra de herramientas



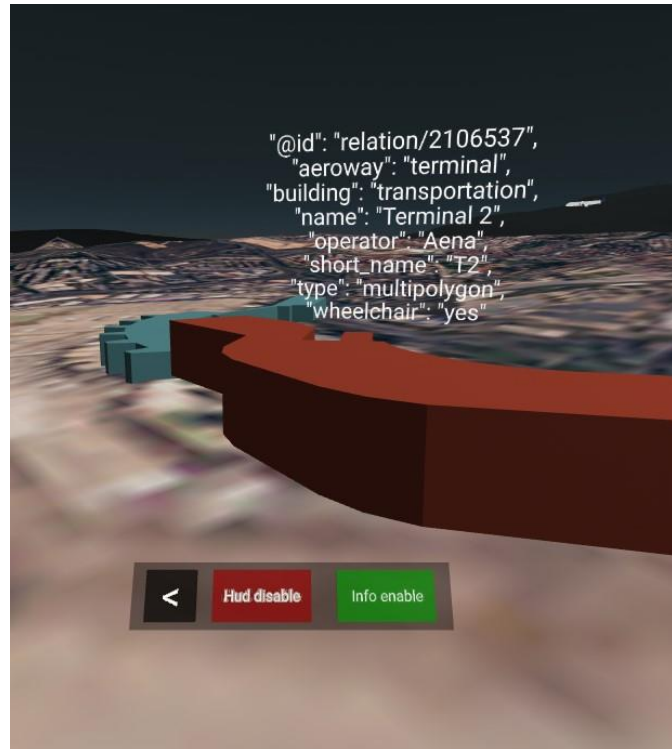
# GUI

## Desplazamiento de componentes HUD



# GUI

## Información contextual





# GUI

## Gestión de altura del usuario

```
invertalEvent: function () {  
    if (this.loaded) {  
        // Obtener la posición actual de la cámara  
        let camPos = this.el.object3D.position;  
        // Posición del padre de la cámara  
        let rigPos = this.el.parentEl.object3D.position;  
        const position = (new THREE.Vector3(0, 0, 0)).copy(camPos).add(rigPos);  
  
        // Obtener la altura del terreno en la posición actual de la cámara  
        const terrainHeight = heightManager.getTerrainHeight(position.x, position.z, false);  
  
        // Actualizar la altura de la cámara para que se ajuste a la altura del terreno  
        this.el.object3D.position.y = terrainHeight + this.personHeight;  
    }  
},  
  
tick: function () {  
    this.throttledFunction();  
}
```

# Construcción de escenas

1. Ficheros DEM (Digital Elevation Model)
2. Usar GDAL para realizar las uniones, recortes, transformaciones y exportación de los binarios.
3. Generación de la capa ráster con Google Earth Engine y el script proporcionado.
4. Descargar los datos de los edificios con Overpass-API.
5. Convertir el fichero de Overpass-API a GeoJSON con el script osmtogeojson.
6. Almacenar datos de vuelos ejecutando openSkyDataSaver.js con node.js (Opcional).
7. Crear el fichero de configuración.
8. Crear página HTML de la aplicación con el fichero de configuración.



Gracias por su atención.

Estoy disponible para  
preguntas.



# Componentes Reutilizables

- Componente de información contextual
- Componente de desplazamiento de entidades HUD
- Barra de herramientas 3D (HUD)
- Geometría edificio

# Trabajos Futuros

- Estimación de la posición a través de las marcas de tiempo.
- Teletransportación con interfaz gráfica dentro del escenario.
- Creación de escenarios de forma dinámica (Concepto teselas)
- Entidades geoespacial configurables por posición wgs84 y modelo glTF e información contextual.
- Interfaz gráfica para búsqueda de vuelos presentes en el escenario.
- Tener varios modelos de glTF para los enumerados de vuelos

## Aircraft category.

- 0 = No information at all
- 1 = No ADS-B Emitter Category Information
- 2 = Light (< 15500 lbs)
- 3 = Small (15500 to 75000 lbs)
- 4 = Large (75000 to 300000 lbs)
- 5 = High Vortex Large (aircraft such as B-757)
- 6 = Heavy (> 300000 lbs)
- 7 = High Performance (> 5g acceleration and 400 kts)
- 8 = Rotorcraft
- 9 = Glider / sailplane
- 10 = Lighter-than-air
- 11 = Parachutist / Skydiver
- 12 = Ultralight / hang-glider / paraglider
- 13 = Reserved
- 14 = Unmanned Aerial Vehicle
- 15 = Space / Trans-atmospheric vehicle
- 16 = Surface Vehicle - Emergency Vehicle
- 17 = Surface Vehicle - Service Vehicle
- 18 = Point Obstacle (includes tethered balloons)
- 19 = Cluster Obstacle
- 20 = Line Obstacle

# Conclusiones

## Aprendido

- Prototipo de herramientas para construcción de escenas configurable y muy expandible.
- Aprendida la tecnología A-Frame para realización de aplicaciones de visualización de datos.
- Conceptos básicos sobre Three.js para la generación de componentes de A-Frame complejos y personalizados.
- Tecnologías GIS como GDAL, Google Earth Engine, OverPass Api y Leaflet.
- Programar con soltura en JavaScript de manera modular.
- Documentar en LaTeX y el uso de herramientas de construcción de diagramas de apoyo como es Draw.io
- Crear servidores de aplicaciones sobre GitHub para mostrar prototipos.

# Construcción De Escenas

## Datos de vuelos

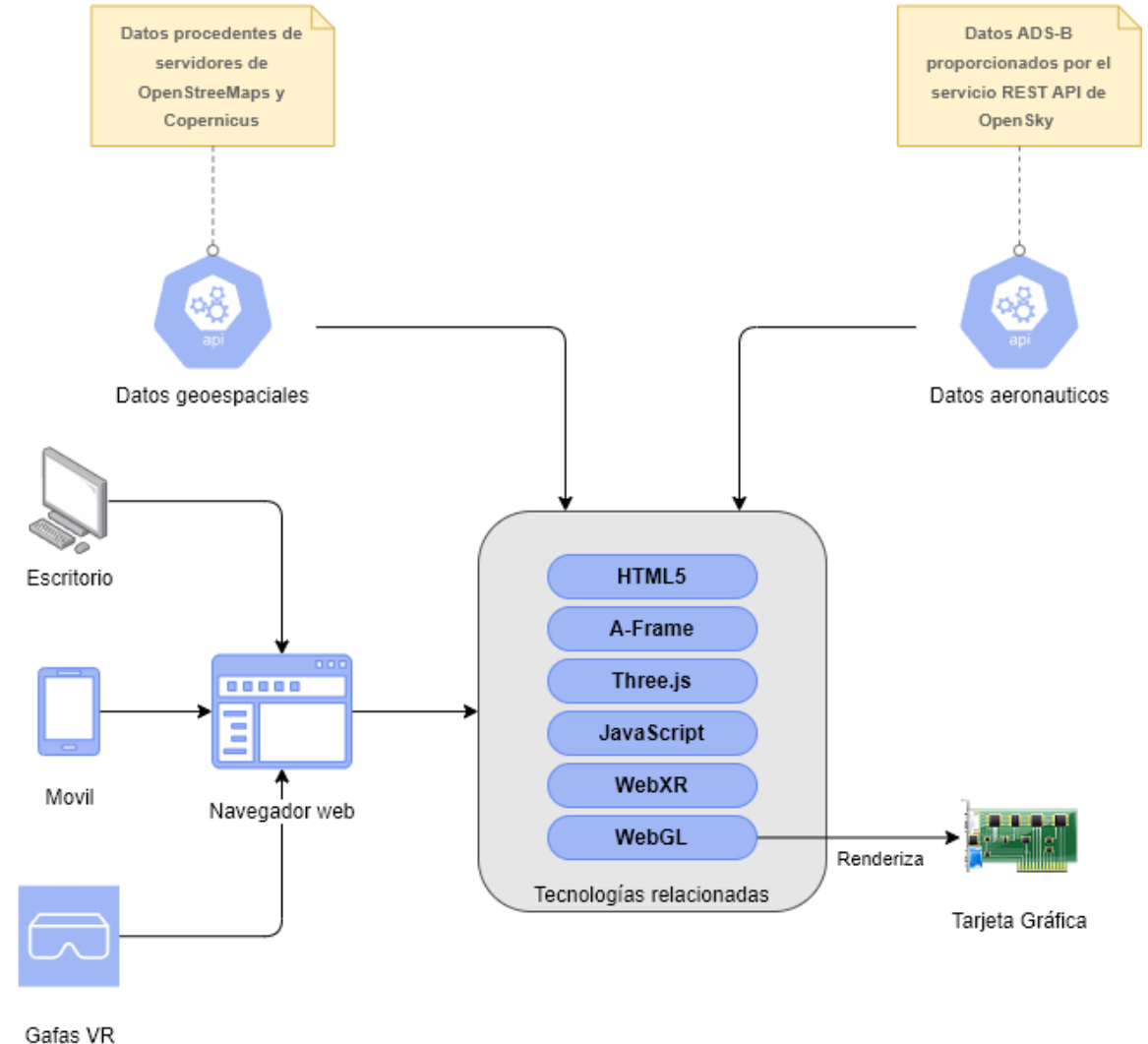
1. Almacenar datos de vuelos ejecutando openSkyDataSaver.js con node.js para ejecuciones de la aplicación en modo asíncrono.

```
import * as configuration from './configuration/configurationModel.js';
import * as openSkyDataSaver from './openSkyDataSaver.js';

//Establecemos las coordenadas del escenario.
configuration.setMerConfig(40.0234170, 40.7441446, -4.2041338, -3.2538165);
//Ruta absoluta a la carpeta donde queremos almacenar los datos de vuelo
configuration.setFlightLocalFolder('C:\\Users\\djpra\\Documentos\\workspaceTFG\\.....');
//Usuario de la API OpenSky si no se posee uno simplemente hay que registrarse.
configuration.setApiUser('xxxxxx');
//Contraseña de la web OpenSky.
configuration.setApiPassword('xxxxxxx');
//Intervalo entre peticiones, recordar que gratuitamente nos e permite vectores de
↔ posición menos a 5seg.
configuration.setDaoInterval(5100);
//Lanzamos el proceso.
openSkyDataSaver.main();
}
```

# Mapa De Tecnologías

- Entornos de ejecución
  - Escritorio
  - Móvil
  - Gafas VR
- Entrada de datos
  - Datos geoespaciales
  - Datos ADS-B



# Contexto y motivación



**1838** Charles Wheatstone comenzó sus estudios de la visión binocular

Estereoscopio: una serie de imágenes planas de un mismo objeto (pero tomadas de diferentes ángulos) que al combinarse y mostrarse individualmente a cada uno de nuestros ojos crean la ilusión de un ambiente 3D dando sensación de profundidad.



**2012** Oculus Development Kit 1

**2020** Oculus Quest 2



**2023** Meta Quest 3



**2023** Apple Vision Pro





# HTML5

- Punto de entrada de la aplicación, es la tecnología que multiplataforma que actúa como lenguaje base para unir las tecnologías de la aplicación.
- Permite crear etiquetas personalizadas (A-Frame)
- Ejecuta JavaScript de forma nativa
- Ejecuta WebGL de forma nativa para renderizar gráficos 3D a través de OpenGL.
- Ejecuta WebXR de forma nativa para crear experiencias de realidad virtual

# Construcción De Escenas

## Generación del fichero de configuración

```
import * as configuration from './configurationModel.js';
//Coordenadas del escenario latmin, latmax, longmin, longmax.
configuration.setMerConfig(40.0234170, 40.7441446, -4.2041338, -3.2538165);
//Coordenadas de la posición inicial del usuario.
configuration.setCamPosition(40.4893, -3.52254);
//Nombre del fichero de edificios sin extensión ubicado en la carpeta
//"data" del proyecto.
configuration.setBuildingFileName('madrid_building');
//sufijo de la carpeta que contiene los vuelos, debe contener el prefijo "flightData"
//Opcional solo para uso offline.
configuration.setFlightLocalFolder('_madrid');
//fichero con la capa raster del terreno ubicado en la carpeta
//"data" del proyecto.
configuration.setMapRaster('Madrid_raster.jpg');
```

# Construcción De Escenas

## Generación del fichero de configuración

```
//fichero binario de alturas para el terreno ubicado en la carpeta
//"data" del proyecto.
configuration.setMapDem('madrid_dem.bin');
//Establece si queremos lanzar la aplicación en modo offline con los datos de la carpeta
//caché.
configuration.setLocalApiMode(true);
//Intervalo de refresco de los datos, cada cuanto se realiza una petición.
configuration.setDaoInterval(2000);
//En caso de modo offline por cual fichero queremos empezar la reproducción.
configuration.setDaoLocalIndex(0);
//Usuario de la API OpenSky si no se posee uno simplemente hay que registrarse.
configuration.setApiUsuer('xxxx');
//Contraseña de la web OpenSky.
configuration.setApiPassword('xxxxx');
```