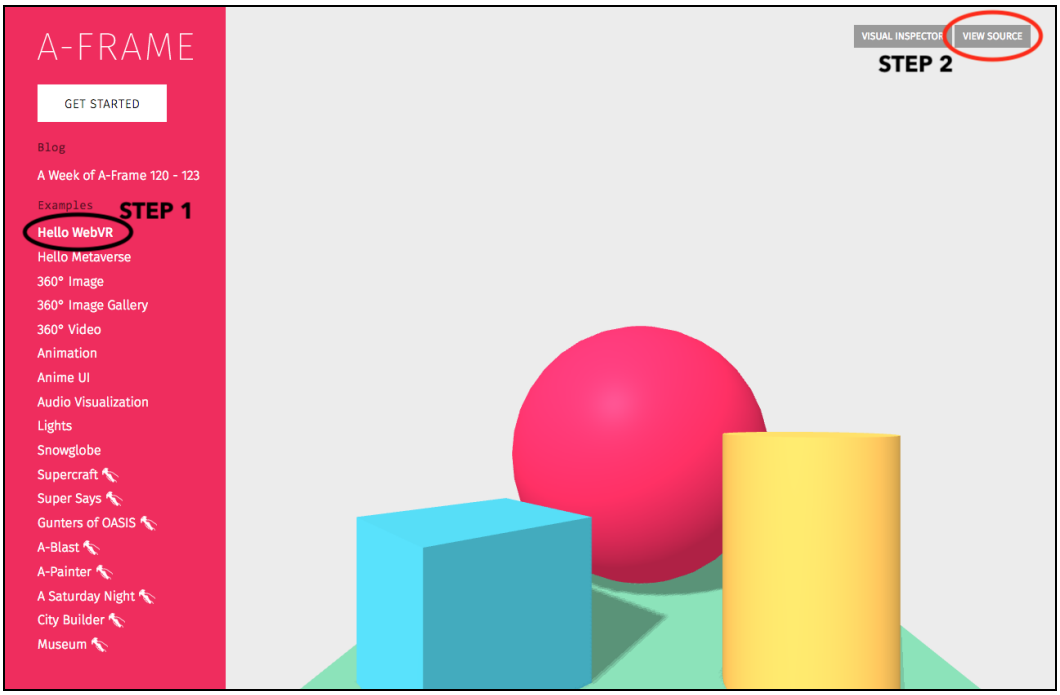
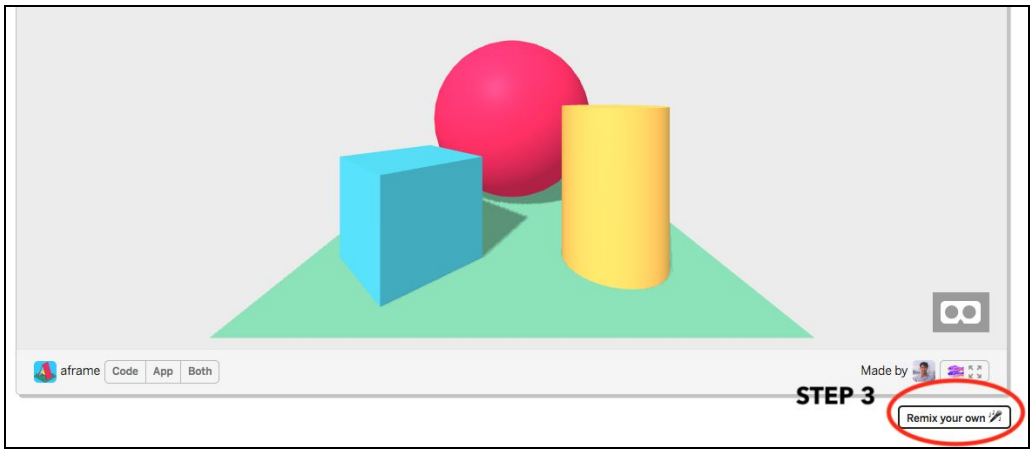
# Augmented Reality using AR.JS

AR.js program is a wrapper that goes on top of A-Frame to use code to integrate augmented reality in a scene. For more information about using A-Frame in conjunction with AR.js to create augmented reality, go to this link aframe.io/blog/arjs/.

When using this type of coding program, using Google Chrome is strongly recommended. To start creating with A-Frame and AR.js, go to the website aframe.io, and click on “Hello WebVR” in the sidebar, then the “View Source” button in the upper-right:




Click on “Remix your own.” This will bring you to glitch.com, which works in collaboration with A-Frame. SignIn/Create an account on glitch.com so you can start to edit the code! (Glitch allows you to create accounts with Facebook, Gmail…)



This should take you to a page that reads “A-Frame Project.” On the side of the webpage there are several tabs that list as “assets, env…” Click the last tab that reads as **index.html**. This is the most important tab, and will be **where we input our code**. The code from the following instructions must be copied into your project in the order that is specified. If confusing, refer to the images (which are an approximation of what the code should look like).

Keep in mind that with code, all “beginnings” have “endings.” When typing in code, something like `<body>` will require you to close the code with `</body>`. This signifies that whatever effect the code has created is being sectioned off. Consider it to be a series of containers within containers. Or when something starts with a quotation mark there must always be an ending quotation mark, for example: `“hiro”` or `src=””`. You will understand the further you go.

# Creating the Base of the Code


Highlight all of the code currently in your index.html tab, and delete it (so that you are left with an empty page). Our instructions will be starting from scratch. First, we must do basic setup and include the latest A-Frame build and AR.js (which will make our A-Frame project augmented reality enabled). Copy the text below into the index.html page on the first few lines.

```
<!DOCTYPE html>
<html>

<script src="https://aframe.io/releases/0.6.0/aframe.min.js"></script>
<script
src="https://jeromeetienne.github.io/AR.js/aframe/build/aframe-ar.js"></script>
</html>
```

Under that, we will define the body. Within the `<body></body>` tags, create a scene using `<a-scene></a-scene>` tags. Adding this scene and embedding AR.js signifies in the code, that we would like to use AR.js to create an augmented reality scene.

```
<!DOCTYPE html>
<html>

<script src="https://aframe.io/releases/0.6.0/aframe.min.js"></script>
<script
src="https://jeromeetienne.github.io/AR.js/aframe/build/aframe-ar.js"></script>

<body style="margin : 0px; overflow: hidden;">
```

```
  <a-scene embedded arjs>
      </a-scene>
</body>
</html>
```

We will now add the marker camera. The marker is essentially an image that will trigger the 3D object to appear in your AR scene. The marker camera will allow the AR object to move according to the marker's position. We will be using the Hiro preset marker in this project. This code should be placed in between the `<a-scene></a-scene>` tags.

```
<body style="margin : 0px; overflow: hidden;">
 <a-scene embedded arjs>
   <a-marker-camera preset= "hiro"></a-marker-camera>

 </a-scene>
</body>
```

**Note** If you would like to create your own custom marker, follow the instructions on this A–Frame blog post: (https://aframe.io/blog/arjs/#customize-your-marker)

Below is the Hiro preset marker we will be using:


Next, we will add a box to our code to act as our AR object. Copy the following code in between the `<a-scene></a-scene>` tags above the `<a-marker-camera></a-marker-camera>` tags.

```
<body style="margin : 0px; overflow:
hidden;">
  <a-scene embedded arjs>
 <a-box position="0 0.5 0"
color="#4CC3D9"></a-box>
  <a-marker-camera preset=
"hiro"></a-marker-camera>
</a-scene>
</body>
```

Your code should look like this:

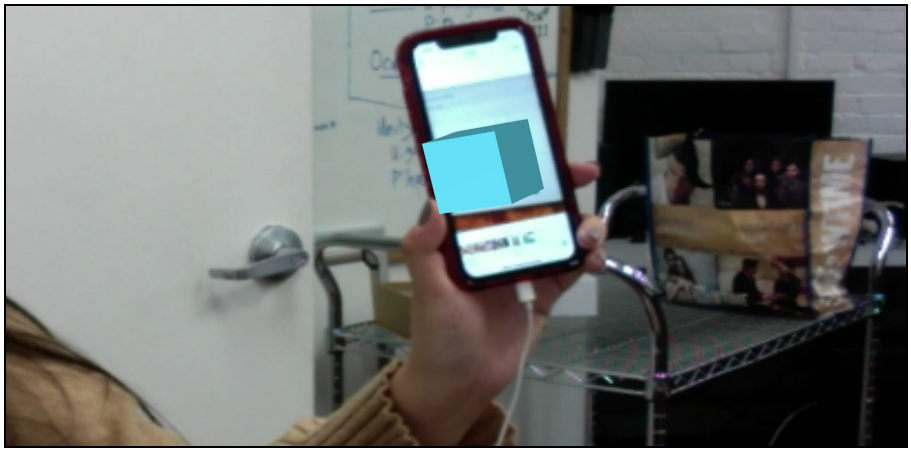
```
<!DOCTYPE html>
<html>

<script src="https://aframe.io/releases/0.6.0/aframe.min.js"></script>
<script src="https://jeromeetienne.github.io/AR.js/aframe/build/aframe-ar.js">
</script>

  <body style="margin : 0px; overflow: hidden;">
    <a-scene embedded arjs>
      <a-box position="0 0.5 0" color="#4CC3D9"></a-box>
      <a-marker-camera preset= "hiro"></a-marker-camera>
```

```
    </a-scene>
  </body>
</html>
```

In the upper left hand corner, there is a “Show” drop down arrow, click it and press “In a New Window.” This will open your completed AR scene in a new window. **The URL of this new window is the custom URL that Glitch has generated for your Glitch project, and you can go to this URL on any mobile or desktop device with a camera, and trigger your AR scene.** Hold the trigger image “Hiro” on the previous page, to the camera of your device, and view your AR cube!


**Whether you are using a desktop or mobile device, make sure that the URL you are typing in is secure.** Regular URLs begin with [http://] but **secure URLs** include an ‘s’**, [https://].** Make sure to include this in the web address when you want to open your custom Glitch URL.

(NOT SECURE) http://vrtruth-newspaper.glitch.me/
**(SECURE)** https://vrtruth-newspaper.glitch.me/



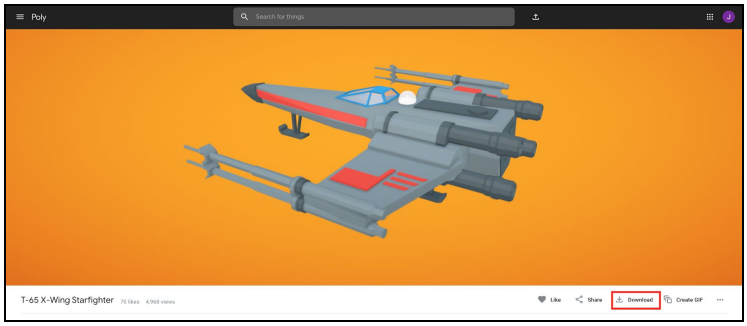
# Downloading Object (OBJ) Models

Now that we have a basic example of creating an AR scene, next we will try inserting 3D models into our project. First **find the following code below in your own index html page, highlight, and delete it**. Because we know the basic setup of an AR scene, we are going to insert a new 3D model into our code, and no longer need the example box.

```
<a-box position="0 0.5 0" color="#4CC3D9"></a-box>
```

First we have to import a 3D model files into our project. Find a model to download on https://poly.google.com. You may also choose any other 3D model site or personal 3D model file you would like to put into AR, as the process is the same. We chose a 3D model of a x-wing

([https://poly.google.com/view/100p3RNw-5Q](https://poly.google.com/view/100p3RNw-5Q)). Below the image of the 3D model, there is a download drop down button. Download the OBJ file.




Once downloaded, open the zip file by double clicking it. A folder will pop up with an OBJ file and an MTL file. The OBJ file is the actual object model. The MTL file is the texture/color/lighting that covers the object. Once you see your OBJ and MTL files have been unzipped, return to your glitch project page. In the same sidebar where your index.html tab is located, there is an **assets** tab. Click on the assets tab.


Drag and drop your OBJ and MTL files onto the page. This ensures you may use them in your project, and they should automatically load and appear like this:

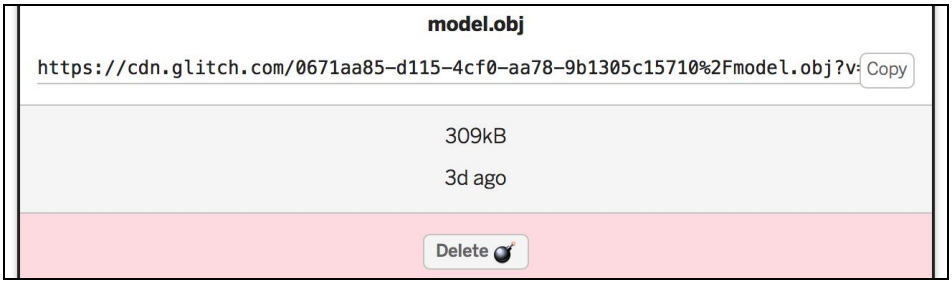

# Adding Object (OBJ) Models

Now that we have our assets loaded onto glitch. We can start to type in the code that will bring our 3D model into AR. Go back to your index.html tab and copy the text below, in between the `<a-scene> </a-scene>` tags.

```
<a-scene embedded arjs>
<a-assets>
    <a-asset-item id="obj" src="url"></a-asset-item>
    <a-asset-item id="mtl" src="url"></a-asset-item>
  </a-assets>

  <a-entity obj-model="obj: #obj; mtl: #mtl"></a-entity>
  <a-marker-camera preset= "hiro"></a-marker-camera>
```

```
</a-scene>
```

Go back to your assets tab. Click on your OBJ file, a smaller window will pop up on the web page with a url.




Copy the url, go back to the index.html page. To put your OBJ file in AR, paste your url into the quotations of the first `<asset-item>`. **DO NOT COPY THE URL FROM THIS WORKFLOW PACKET**. Because everyone will be using different files, urls will vary person to person, simply copy and paste the url from your assets.

```
<a-scene embedded arjs>
<a-assets>
    <a-asset-item id="obj"
src="https://cdn.glitch.com/0671aa85-d115-4cf0-aa78-9b1305c15710%2Fmodel.obj?v=
1571160407881"></a-asset-item>
    <a-asset-item id="mtl" src="url"></a-asset-item>
  </a-assets>

  <a-entity obj-model="obj: #obj; mtl: #mtl"></a-entity>
 <a-marker-camera preset= "hiro"></a-marker-camera>
</a-scene>
```

Now do the same for your MTL file. Go back to your assets tab. Click on your MTL file, and a smaller window will pop up on the web page with a url. Copy the url, go back to the index.html page. Paste your url into the quotations like below. This will allow your MTL file to appear on your 3D model

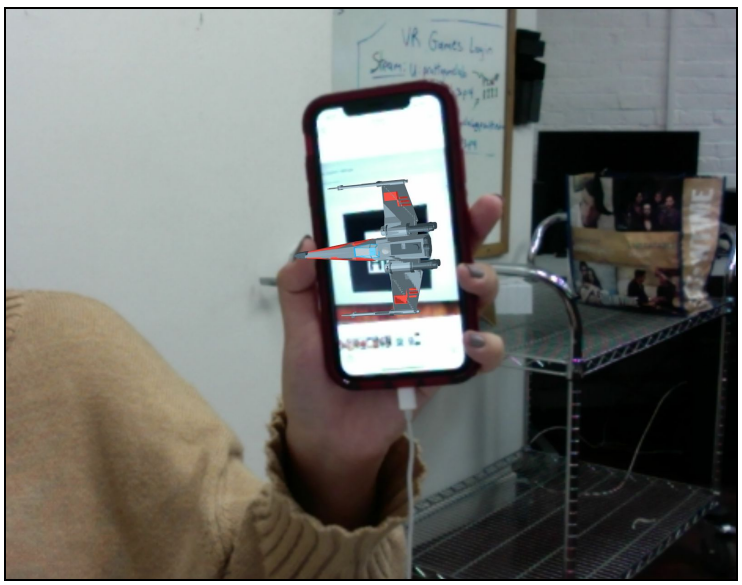
```
<a-scene embedded arjs>
<a-assets>
    <a-asset-item id="obj"
src="https://cdn.glitch.com/0671aa85-d115-4cf0-aa78-9b1305c15710%2Fmodel.obj?v=
1571160407881"></a-asset-item>
    <a-asset-item id="mtl"
src="https://cdn.glitch.com/0671aa85-d115-4cf0-aa78-9b1305c15710%2Fmaterials.mt
l?v=1571160523644"></a-asset-item>
  </a-assets>

  <a-entity obj-model="obj: #obj; mtl: #mtl"></a-entity>
 <a-marker-camera preset= "hiro"></a-marker-camera>
</a-scene>
```

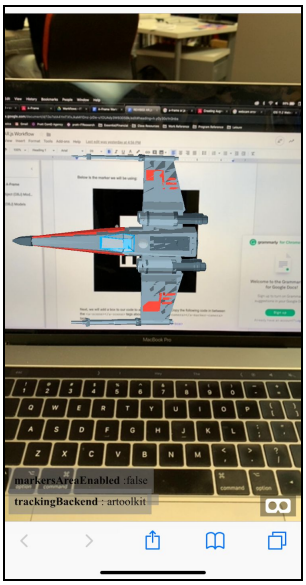
Your code should look like this. Again, the url will vary from person to person):

```
<!DOCTYPE html>
<html>

<script src="https://aframe.io/releases/0.6.0/aframe.min.js"></script>
```

```
<script
src="https://jeromeetienne.github.io/AR.js/aframe/build/aframe-ar.js"></script>

<body style="margin : 0px; overflow: hidden;">
  <a-scene embedded arjs>

  <a-assets>
    <a-asset-item id="obj"
src="https://cdn.glitch.com/0671aa85-d115-4cf0-aa78-9b1305c15710%2Fmodel.obj?v=
1571160407881"></a-asset-item>
    <a-asset-item id="mtl"
src="https://cdn.glitch.com/0671aa85-d115-4cf0-aa78-9b1305c15710%2Fmaterials.mt
l?v=1571160523644"></a-asset-item>
  </a-assets>

  <a-entity obj-model="obj: #obj; mtl: #mtl"></a-entity>

    <a-marker-camera preset= "hiro"></a-marker-camera>
  </a-scene>
</body>

</html>
```

In the upper left hand corner, click the “Show” drop down arrow, click it and select “In a New Window.” This will open your completed AR scene. Hold the trigger image (“Hiro” on the previous page) to the camera of your device, and view your augmented reality scene.






You can also use this code on your phone. Just go to your project page on your mobile device and press the “Show” drop down arrow, click it and press “In a New Window.” This will open your completed AR scene in a new window. Hold the trigger image up to the camera of your iPhone or Android to view your AR scene on your phone. For **IOS, users should use Safari**, as Google Chrome does not support the code. **Android users may use Google Chrome.**

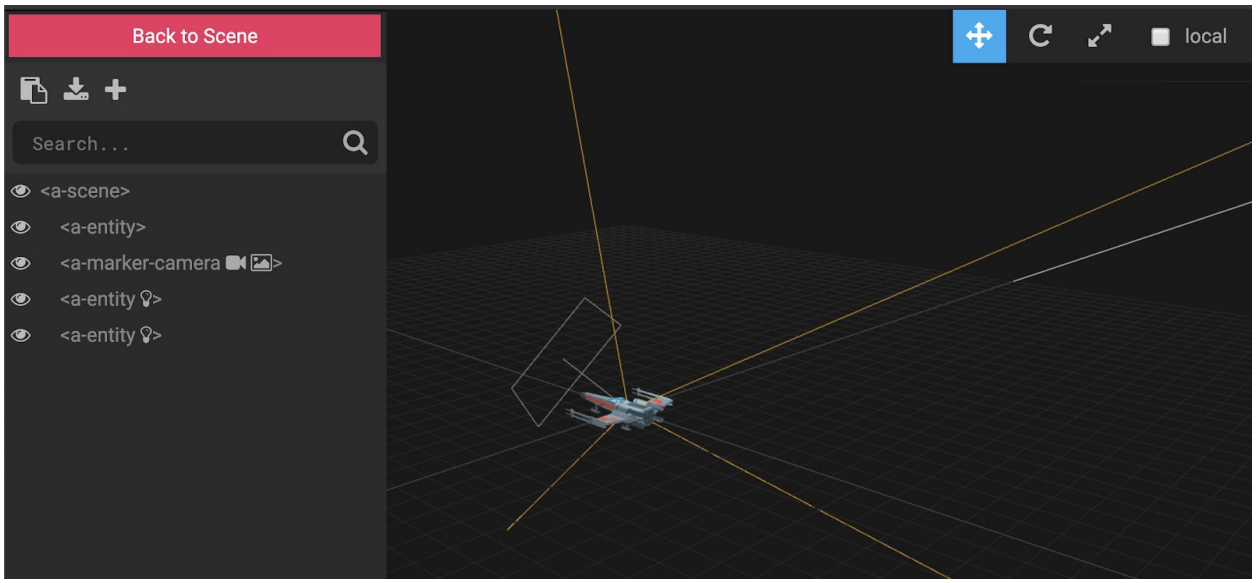
# Editing Model Position, Rotation, Angle, Scale, Etc.

If you would like to edit attributes of your OBJ model such as position, rotation, angle, or scale, you can either manually do this in your html tab, or use the Inspector tool A-Frame provides. If you would like to do this manually in your code, you can enter such attributes in between your <a-entity></a-entity> tags.

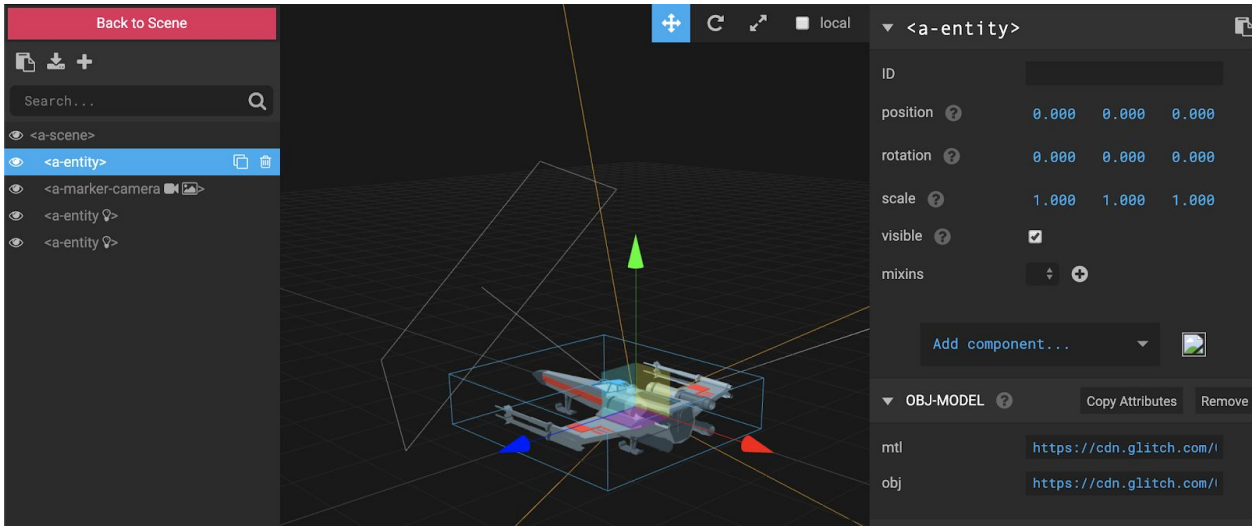
```
<a-entity obj-model="obj: #obj; mtl:#mtl" position="0 0 -10" scale="3 3 3"
rotation="0 100 0" scale="5 5 5"></a-entity>
```

You can manually change the x, y, and z, properties of your 3D model by adding attributes such as 'position', along with values to those attributes such as "10 0 0"' This will indicate to the program that you would like the position of your OBJ model to be 10 unit away from your trigger image (the center or "0 0 0") of your AR scene, on the x axis. The object will then appear to your 10 units to your right when you trigger your AR scene, instead of directly on your marker.

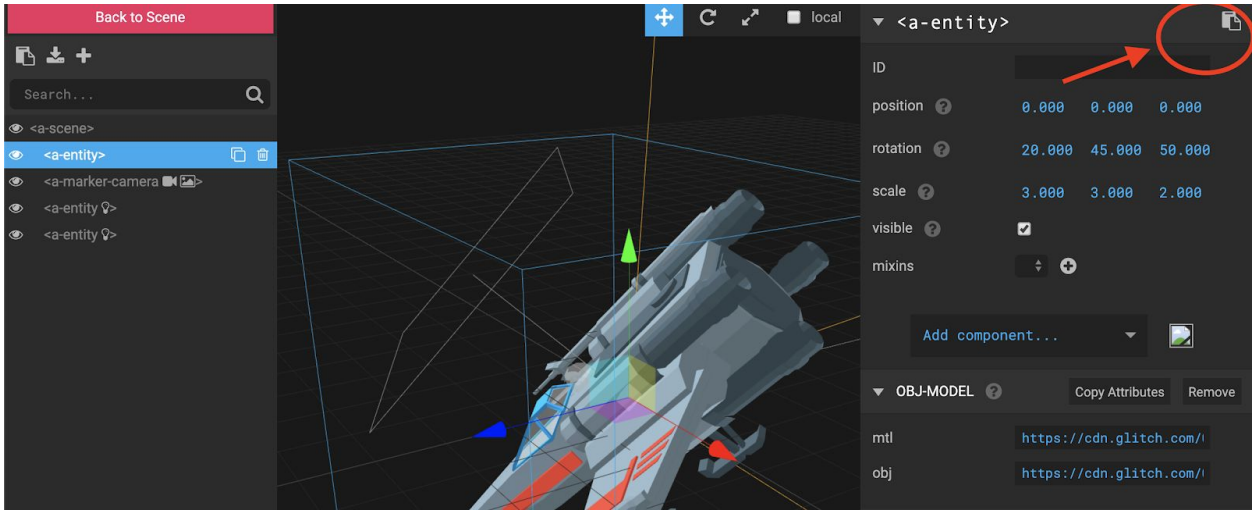
Using the Inspector tool A-Frame provides is another way of editing these same attributes of your OBJ model. You can do this by clicking the "Show" drop down button in the upper left corner of your html page, and selecting "In a New Window." This will open your completed AR scene in a new window. If you are on a **Mac device, hold down Control + Option + i**. If on a **Windows device, hold down Alt + i**.




Your page will load a gridded space with your OBJ model appearing. If you look to the left side of the page, a tab is open, with several different "tags" to choose from. Clicking on different tags will allow you to visually edit different aspects of your code. Clicking on your OBJ model, which will appear in the left side tab as **<a-entity>** in the Inspector tool, will allow another tab to open to the right, with attributes such as position, rotation, and scale being editable.




You can play with editing the values of these attributes, and the OBJ model will change to accomodate any changes to its attributes. It is important to note that any **EDITING DONE IN THE INSPECTOR TOOL IS NOT AUTOMATICALLY SAVED.** It is important to know that any work you do editing in the Inspector Tool is not automatically saved, and is not automatically applied to your code. The Inspector tool is a tool to help you visualize your OBJ model, but it does not automatically change the code, but you can do that.




In the upper right hand corner of the webpage, there is an icon of a clipboard with a piece of paper on top that has a folded corner. Click this icon. This icon creates an automatically creates an entirely new line of code that can be copied and pasted into your existing html tab.

Go back to your code in Glitch in your html tab, and highlight the existing code with the tags <a-entity></an-entity>, and delete it. In the same line/place paste the new code that was copied from clicking the clipboard icon, and it will paste the code to your new attributes!

# Fine-tuning & Troubleshooting your Code

If you are experiencing **issues with loading your augmented reality scene onto your phone**, there are a few tricks that may resolve your frustrations. If, when loading the url onto your phone, you receive a pop up notification that says “Webcam Error…” or something similar, you can try the following solutions that will likely resolve your issues.

1. First, recheck your url. Make sure you have typed it in correctly, that nothing is misspelled, misplaced, or missing.

2. Your phone may not have given permission for glitch.com to activate and use your phone’s rear camera. If your privacy settings are like this, you may have to go into your phone settings and allow the browser (Safari, FIrefox, Google) to be allowed to use it.

3. If it is not an issue of privacy/permission, it may simply be an issue of the website url. Make sure that when you open your AR scene on your phone, that the website is opened securely. You will be able to see this by looking at the url (examples below). Like below, if your url begins with “https://” it is secure and will open properly. If your url begins with “http://” it is not secure. Secure websites will also often have a closed “lock” icon that comes before the url.

   **(NOT SECURE)** http://glitch.com
   **(SECURE)** https://glitch.com/

If you are experiencing **issues with loading your augmented reality scene onto your desktop device**, there are a few tricks that may resolve your issues.

1. First, recheck your code. Make sure quotations are at both ends of a url, that the url was pasted in correctly (in the corresponding tag), that opening tags have closing tags, that your containers are all in order.

2. If your show page is gray and blank, your code is fine. The blankness of the page simply means that you are not holding your marker up to your camera. And because your camera is not registering your marker, the window is gray and blank. Simply hold your marker properly, up to the camera so that it may register the marker, and your augmented reality scene will appear.

3. If your page simply will not appear or reflects blank white space, no matter how you position and hold the marker up to the camera, this may signify an issue with the browser you are working in. We recommend **Google Chrome for desktop devices, laptops, and Androids**, but **Safari for IOS** devices. If your desktop device is not registering the AR scene on Chrome despite the marker, it may be due to your version of Chrome. Check to make sure that your desktop device is up to date with the latest version of Google Chrome. If it is not, update your browser.

4. If you have attempted all these solutions and they haven't worked, try opening the website/code using Mozilla FireFox. Using a new browser with different features may help or resolve the situation. Though **we do not recommend using Safari** on desktop for this type of coding.

If you are experiencing **issues with loading your assets**, there are a few tricks that may resolve your frustrations. First make sure your url is copied pasted into the the correct space in the correct way: check for missing parts of the urls, for double pasted urls, and that the correct asset urls are pasted in the correct tag (the obj url in the obj <a-asset-item> line and the mtl url is pasted into the mtl <a-asset-item> line).

1. If all your code is in order, one concern maybe that your asset is simply not compatible with Glitch. To test whether or not it is an issue of the asset, try replacing your asset with the X-Wing asset (https://poly.google.com/view/100p3RNw-5Q). Through multiple tests, this asset has proven to work with the Glitch program, so if this X-Wing functions within your code, it maybe that your original 3D asset does is not compatible with Glitch
    a. Certain 3D assets are either too complex or simply not compatible with the Glitch program. This cannot really be helped beyond changing your 3D asset to something else that is compatible
2. In the case that your OBJ file might show up in the AR scene, but the MTL (material) may not appear (causing your 3D assets to appear blank white), this maybe because your MTL file is too complex.
3. If your camera recording is appearing on your screen with the marker, but your AR 3D asset is not being triggered, try moving your marker around the recorded space.
    a. Remember that in order for the AR scene to be triggered, your camera needs to be able to read your marker. If you see your marker on the screen, but nothing is being triggered, experiment in the position and distance at which you are holding your marker.

Unfortunately, code, by nature, can sometimes be finicky. It may seem like a hassle to have to troubleshoot solutions many times, but don't worry. Coding is often just about double checking your code to make sure it is correct and finding logical solutions to malfunctions. As it is in every other profession, finetuning and troubleshooting is just another part of creating something amazing!

# Supplemental Links

More on using A-Frame in with AR.js for AR
https://aframe.io/blog/arjs/

A–Frame WebVR link
https://aframe.io

Creating a Custom Marker
https://aframe.io/blog/arjs/#customize-your-marker
https://medium.com/arjs/how-to-create-your-own-marker-44becbec1105

Using Multiple Markers
https://aframe.io/blog/arjs/

Google Poly 3D Models
https://poly.google.com
X-Wing: https://poly.google.com/view/100p3RNw-5Q

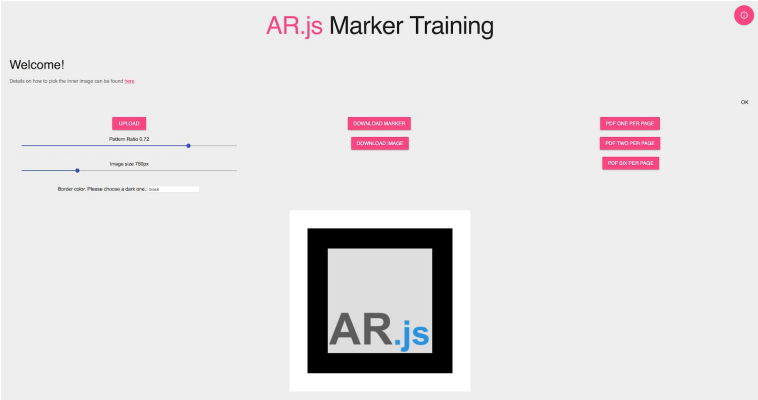
# Creating Custom Markers (Trigger Images)

The below is an example of what your code should look like right now. Notice the yellow highlighted portion of the code. This is, as we've mentioned before, is your marker camera. The marker is essentially an image that will trigger the 3D object to appear in your AR scene and allow the AR object to move according to the marker's position. We have previously been using a preset marker in this project, but custom markers can also be made for your own use!

```
<!DOCTYPE html>
<html>

<script src="https://aframe.io/releases/0.6.0/aframe.min.js"></script>
<script
src="https://jeromeetienne.github.io/AR.js/aframe/build/aframe-ar.js"></script>

<body style="margin : 0px; overflow: hidden;">
  <a-scene embedded arjs>

  <a-assets>
    <a-asset-item id="obj"
src="https://cdn.glitch.com/0671aa85-d115-4cf0-aa78-9b1305c15710%2Fmodel.obj?v=
1571160407881"></a-asset-item>
    <a-asset-item id="mtl"
src="https://cdn.glitch.com/0671aa85-d115-4cf0-aa78-9b1305c15710%2Fmaterials.mt
l?v=1571160523644"></a-asset-item>
  </a-assets>

  <a-entity obj-model="obj: #obj; mtl: #mtl"></a-entity>

    <a-marker-camera preset= "hiro"></a-marker-camera>
  </a-scene>
</body>

</html>
```

Below is the site where you can generate your own marker to use to trigger your AR scene. Notice the "UPLOAD" button in the upper left hand corner of the site. This is where you will upload your inner image. You can upload any image you want to use as a marker, but keep in mind that using black and white colors, or colors with bold and clearly defined patterns/logos, will be easier for the program to register and trigger. While images without clear lines and definition, such as photos or intricate illustrations, will be difficult for the program to pick up. Go to the link below:
https://jeromeetienne.github.io/AR.js/three.js/examples/marker-training/examples/generator.html

Once you upload your image, it will appear within a black frame, and you can use the toggles below the “UPLOAD” button to determine the thickness of the frame and the resolution of your trigger image. A preview image of your marker should be generated onto the screen.




Next, download the marker. Click the top middle button on the site “DOWNLOAD MARKER” to download your complete marker. This is not the actual marker image, this downloads the code that pairs with your marker. Then, download your marker image by clicking the “DOWNLOAD IMAGE” button under the “DOWNLOAD MARKER” button. On the right side of the site, you will see there are a few other options for downloading your marker, this is for using multiple marker triggers in one scene. For now, just download the image.


The center image may be different, but your marker should look something like the above photo: a white outer frame, black inner frame, and an image inside the two. Like the assets you have dragged and dropped into glitch before, do the same with the marker image. Drag and drop the file into your assets tab.

We’ve previously established that the below yellow highlighted text is our preset marker. SInce we are going to be using a custom marker, highlight this line of code in your own html tab, and delete it from your code.

```
<a-marker-camera preset= "hiro"></a-marker-camera>
```

In the same place/line where you’ve deleted the above code, replace it with the code below.

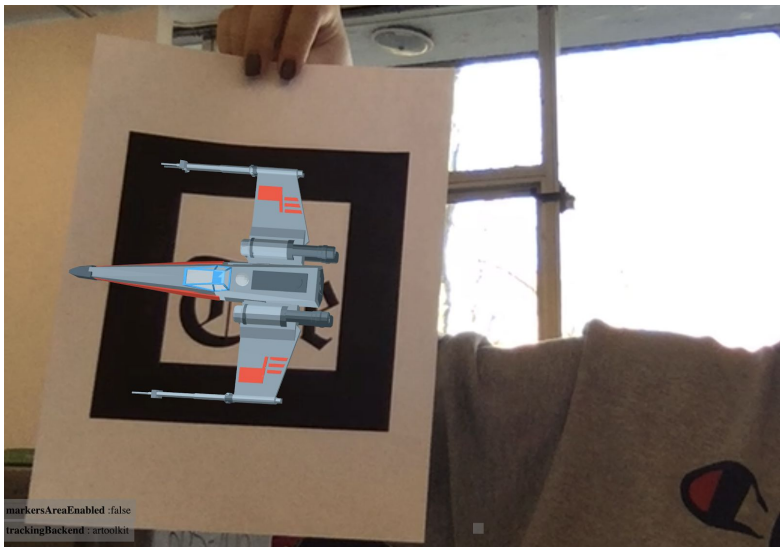
```
<a-marker-camera type=”pattern” url=””></a-marker-camera>
```

Like we have with the assets before, go to your assets tab, and Copy + Paste the url of your marker image. Paste the url in between the quotations of ( url=”” ). This will allow the program to know to use your custom marker as a trigger for your AR scene!

```
<a-marker-camera type=”pattern”
url=”https://cdn.glitch.com/88846d9a-8534-415f-a918-fc842e843364%2FMarkerImage.
png?v=1580750670866”></a-marker-camera>
```

Below is an approximation of what your code should look like. Though the url’s maybe different, all the tags should have a beginning and end, and be enclosed within each other. If you have a red dot appear next to any of your lines of code, this is an indication that there is a technical error in your code. Though the only part of the code that has changed, is highlighted in yellow.

```
<!DOCTYPE html>
<html>

<script src="https://aframe.io/releases/0.6.0/aframe.min.js"></script>
<script
src="https://jeromeetienne.github.io/AR.js/aframe/build/aframe-ar.js"></script>

<body style="margin : 0px; overflow: hidden;">
  <a-scene embedded arjs>

  <a-assets>
    <a-asset-item id="obj"
src="https://cdn.glitch.com/0671aa85-d115-4cf0-aa78-9b1305c15710%2Fmodel.obj?v=
1571160407881"></a-asset-item>
    <a-asset-item id="mtl"
src="https://cdn.glitch.com/0671aa85-d115-4cf0-aa78-9b1305c15710%2Fmaterials.mt
l?v=1571160523644"></a-asset-item>
  </a-assets>

  <a-entity obj-model="obj: #obj; mtl: #mtl"></a-entity>

<a-marker-camera type="pattern"
url="https://cdn.glitch.com/88846d9a-8534-415f-a918-fc842e843364%2FMarkerImage.
png?v=1580750670866"></a-marker-camera></a-scene>

</body>

</html>
```




Discover your Augmented Reality scene using your custom marker! In the upper left hand corner, click the “Show” drop down arrow, click it and select “In a New Window.” This will open your completed AR scene. Hold your custom trigger image (preferably printed out) up to the camera of your device, and view your augmented reality scene.

If you encounter any issues, refer back to the **Fine Tuning & Troubleshooting page**. Remember that in order for the AR scene to be triggered, your camera needs to be able to read your marker. If you see your marker on the screen, but nothing is being triggered, experiment in the position and distance at which you are holding your marker.