AI CUBED HOLIDAY PROGRAM ‘22

INTRODUCTION TO THE METAVERSE

DRAGON REALM VR

PART II

Designed and prepared by Dion Stojsavljevic

1. Advanced Elements

We now have a great looking castle, complete with textures and some of your own color and style.

We also have added some elements to our world to start making it a bit more dynamic.

Now we want to start adding detail that really makes our world come alive, and this includes animation.

1.1 Drawbridge

No castle is complete without a drawbridge. Especially since we have a moat!

We will start with a box tag, and add the following properties:

- `Position 1 1 -6.2`
- `Width 2`
- `Height 4`
- `Depth 0.2`
- `Add the texture` www.ai3.academy/inspire/metaverse/drawbridge1.jpg

6.2 Adding Animation to objects

Animation is added the same way a property is. Below you will see a complete example of the drawbridge tag, complete with properties and animation

- ***Note the layou***t – it's very clear to see all the animation properties as each one is on a new line. Make sure you use this format otherwise debugging is very difficult!

```
<a-box position="1 1 -6.2"
       width="2" height="4" depth="0.2" src="drawbridge1.jpg"

      animation__1="
      property: rotation;
      from: 0 0 0;
      to: 90 0 0;
      dur: 4500;
      dir: normal;
      "
      >

</a-box>
```

WHAT to animate

- Position
- Rotation
- Color
- Opacity

From/To

Starting and ending position of our animation

Dur

How long to animate for

Other useful properties: **loop** (value or true for infinite), **dir** (normal, reverse or alternate)

What did your animation do? The door should have rotated on its center axis automatically, but it doesn't look right yet…. So we need to add a second animation that runs at the same time.

After the first property, add this additional property:

```
      animation__2="
      property: position;
      from: 1 1 -6;
      to: 1 0 -4.7;
      dur: 4500;
      dir: normal;
```

We are now moving position AND rotation at the same time.

Notice that the from and to are different for both animation types.

The drawbridge should move much more naturally now!

Useful Fact on colors **- you can only use RGB format for animating from and to colors!**

6.3 Adding Pre-Made GLTF models

A glTF™ (GL Transmission Format) is a royalty-free specification for the efficient transmission and loading of 3D scenes and models by engines and applications. **glTF** minimizes the size of 3D assets, and the runtime processing needed to unpack and use them.

More info: https://www.khronos.org/gltf/

In simple terms, they are awesome 3D images that you can download and add into your work for free! Sign me up!!

The image to the right is actually a GLTF rendered model, not a real car!!

First, we are going to add an image that's already been prepared for you to understand the tags to use. The new tag we are going to use is an assets tag:

```
<a-assets> </a-assets>
```

- Create these tags under the section called 'Shark'
- Place them on different lines, and in between them, create another new tag as follows:

```
<a-asset-item> </a-asset-item>
```

The first tag tells our browser that we will be loading in some new assets.
The second one is used for each specific asset, or file.

Add the following properties:

- `id="shark"`
- `src=" "`

IMPORTANT UPDATE!

You will need to follow the instructions in the next **section 6.4 Uploading your own GLFT files** to get the correct url for your shark**.**

Download your shark file here first:

http://www.ai3.academy/inspire/metaverse/Shark.glb

All we have done so far is set up the file ready for use, next we add it.

You will notice we set up an ID called “shark” in the asset item tag. Now we will reference that ID in a new tag called entity:

```
<a-entity>  </a-entity>
```

Add this in right after the </a-assets> tag and then add the following properties:

- `gltf-model="#shark"`
- `position="4.5 -0.5 -4.1"`
- `scale="3 3 3"`
- `rotation="-15 -50 0"`

Now run your code….

Our shark is sad because we still can’t see it….

It IS there, but if you look at the Y position above, we have placed it just below the water level.

- Try and change the height to 1 and see what happens?

Hopefully you can now see our shark making a spectacular jump out the water!

Let’s add an animation to get our shark moving. Add the following property (don’t forget to reset the sharks position!):

```
animation__1="

 property: position;
 from: 4.5 -0.5 -4.1;
 to: 4.5 0 -4.1;
 dur: 4000;
 dir: alternate;
 loop: true;
 "
```

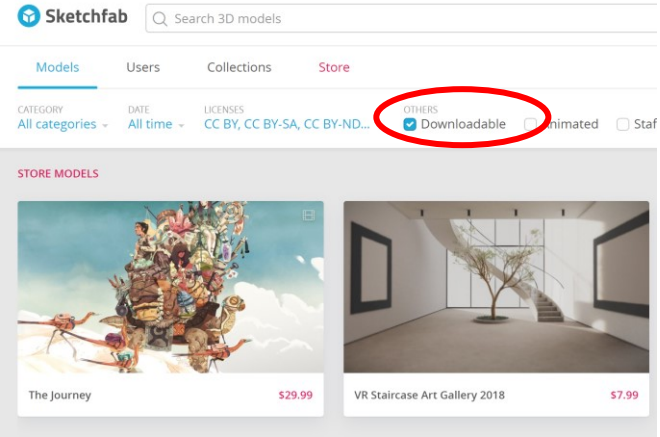
Now we should have a shark lurking beneath the water and popping up every 4 seconds!

6.4 Uploading your own GLFT files

First up, select and Download a free GLTF model from your favourite free resource site. You can do a search on google…

I recommend https://sketchfab.com/

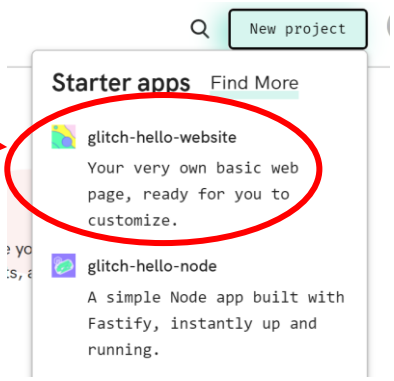
If using sketchfab, select downloadable models only, and be look out for free ones!



Once you have downloaded your model, you will need to upload it to a hosting service, as you cannot run them from your local PC when using the Live Server extension on Visual Studio.

We will use a great service called Glitch to upload our model. Here are the steps:

1. Create Glitch account at: https://glitch.com
2. Note: an email address is required to create an account
3. Create new web project
4. Sellect the ***"Asset"*** button from the left menu
5. Upload glft model using ***"Upload an Asset"***
6. When the file loads, click on it to get the URL
7. Paste this url into the "src" section of your asset-item tag
8. Add your properties and you are ready to go!



NOTE: Some GLTF files are not compatible with some versions of the JavaScript libraries we are using. If it doesn't work, try another file, or another library version (ask your teacher about this!)

Remember the story you wrote right at the start of this project?

Now's your chance to add some of your own elements to bring your story to life!

6.5 Setting the Camera

The camera is your view of the world in VR. By default, it starts at position 0 0 0.

Let's start by making a small adjustment to where the camera starts when you run your code, to make the starting point a bit more interesting.

Find the comment towards to top of your HTML file called 'set the camera position', and under it place the following code:

```
<a-entity position="-2 0 2"> </a-entity>
```

Run your code again. Can you see the difference?

There is another reason we want to change the camera angle, and that is so we are not starting by directly looking at the drawbridge, because it's time to make it only open when we want to!

6.6 Interacting with Objects – Gaze-based Interaction

When we stare at something in VR, we call this our **gaze.**

We can interact with objects in VR in a number of ways – by clicking objects with a button, or by simply staring at them with our gaze.

Since our Cardboard VR headsets have a lack of buttons, we will be focusing on gaze based interaction.

In between the last entity tags you created, add the following:

```
<a-camera>
        <a-cursor fuse="true" fuse-timeout="2000">
        </a-cursor>
</a-camera>
```

The above code adds 2 important things:

1) ***A cursor*** – a point on our screen that we can aim and point at objects
2) ***A fuse*** – this is a countdown timer for how long we need to look at an object before it is triggered. In this example, the fuse is set for 2000, or 2 seconds.

When you run your code now, you should see the cursor pointer.

Let's now go and make a small modification to our draw bridge so it can use the new interaction.

Find your drawbridge code, and go to the animation properties, shown below.

- Can you see what has changed below?

```
animation__1="
property: rotation;
from: 0 0 0;
to: 90 0 0;
dur: 4500;
startEvents: click;
             "
animation__2="
property: position;
from: 1 1 -6;
to: 1 0 -4.7;
dur: 4500;
startEvents: click;
              "
```

We have now removed the normal animation playback that started automatically and replaced it with a start events property.

While it says 'click', because we have already set a 'global' fuse, that means all clicks in our code are triggered by our gaze.

6.7 More Camera controls

Depending on how we design our VR world, we can set up our controls to work in different ways. Try the following to experiment on how it changes your controls, both with the mouse and in VR mode.

Modify the entity tag you added on the last page with new properties:

```
<a-entity position="-2 0 2" camera wasd-controls look-controls>
</a-entity>
```

More information can be found here:

https://aframe.io/docs/1.2.0/components/look-controls.html

6.8 Adding Environmental Effects

There are many different ways we can add effects to our world, some a lot easier than others!

As all our commands link back to a JavaScript library, some helpful people have already created some scripts to help us. That is the great thing about an open-source community.

Remember your story… what kind of weather did it have?

Lighting

Look for the ambient lighting tag near the top of your code, and use an entity tag with the following properties:

- `id="light"`
- `light="type: ambient;`
- `color: #888"`

You can experiment with different colors, and also with the following properties:

Try these other light types:

- directional
- hemisphere
- point
- spot.

There is also an additional property called “Intensity” that accepts a number value.

How dark mode users describe light mode:

For more properties and info on lighting use, see here:
https://sodocumentation.net/aframe/topic/10078/light--component-

Rain


A beta version A-Frame rain

Adding rain is very similar to the way lighting works.

Add an entity tag again under the rain comment, and use the following properties:

- `id="rain"`
- `particle-system="preset: rain; color: #24CAFF; particleCount: 5000"`
- Note that the above is all one line

You can experiment with the particle count, or if you would like a different looking rain effect, here is an alternate rain library:

https://github.com/takahirox/aframe-rain

6.9 Animating the Camera

As our cardboard headsets don't have any controls for moving around our world yet (it only simulates moving your head an looking around), we may want to add a path for our camera to move.

Below is the code you already have for the camera, with an animation property added.

```
<a-entity
    position="-2 0 2" camera wasd-controls look-controls
    animation="property:position; to: 0 3 0; dur: 10000">

        <a-camera>
              <a-cursor fuse="true" fuse-timeout="2000"></a-cursor>
        </a-camera>

 </a-entity>
```

***WARNING:* Moving the camera can made people viewing your world nauseous! Only move at a very slow speed!!**

END OF PART II

APPENDIX A: Dragon Realm VR – A-Frame Quick Reference guide

Item	Tag Syntax
Comment	<!-- *comment text goes here* -->
Floor (Plane)	
Cylinder	
Box	
Cone	
Properties	
Properties	
Sky	
Textures	
Circle	
Ring	
Animation	
Assets	
Asset Item	
Entity	
Cursor	
Camera	