AI CUBED HOLIDAY PROGRAM ‘22

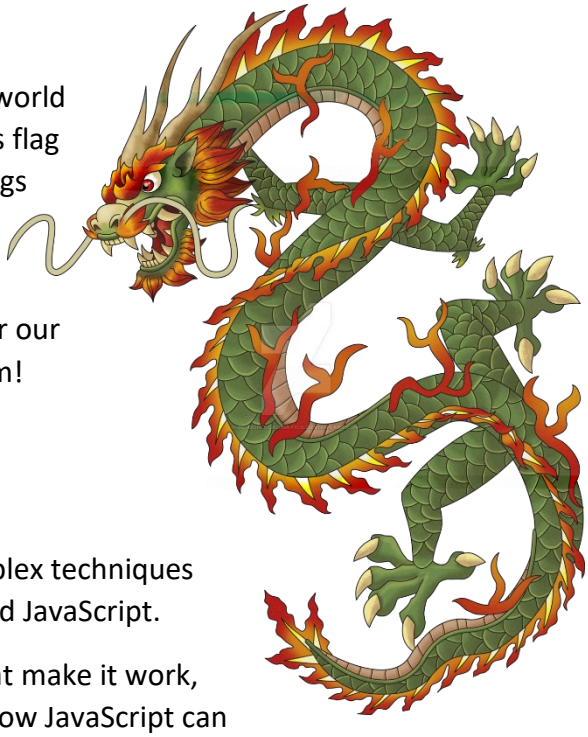
INTRODUCTION TO THE METAVERSE

DRAGON REALM VR

PART III

Designed and prepared by Dion Stojsavljevic

7. Enter the Dragon….



Brave adventurer, so far you have built a world with a mighty castle, with your kingdoms flag flying proudly, a bailey for royal gatherings and a drawbridge and moat to protect your castle from many dangers.

Since our world is called Dragon Realm, it's time for our final addition and introduce a dragon into the realm!

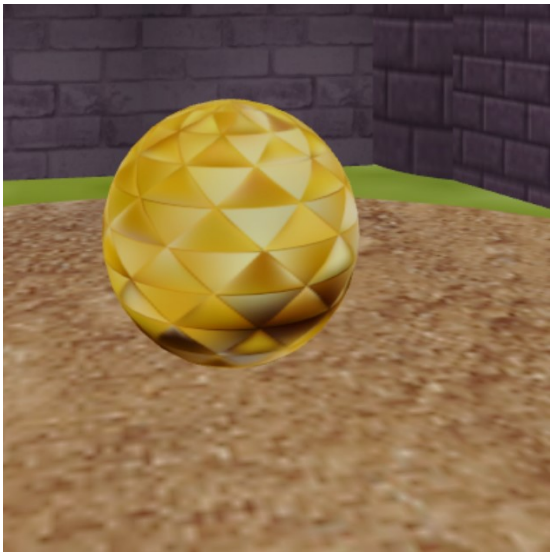
A royal script…

For our final quest we will look at some more complex techniques of how to interact with elements using A-Frame and JavaScript.

We won't be able to explain all of the elements that make it work, but it will be enough of a start that will show you how JavaScript can work within a HTML and A-Frame document.

To start, we will introduce a *mystical floating orb* to your world, that within holds the power to unleash a dragon!

We will be building a sphere with animation, adding a GLTF object with three animations, and a trigger that triggers one animation by looking at another! Phew….



Let's get started…..

7.1 Creating the mystic orb

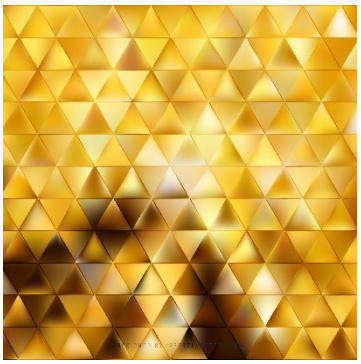
We will start by creating a new shape – a sphere:

```
<a-sphere> </a-sphere>
```

Now let's add some properties to make it appear in the middle of the castle bailey.

Note: You can move it later if you want to put something else in the middle of your castle or add more orbs to your world once you understand how it works.

- position="1 1 -11.1"
- radius="0.30"
- src="orb.jpg"



You can download the orb texture like the one pictured from our example at the link below, or feel free to use your own: http://www.ai3.academy/inspire/metaverse/orb.jpg

Next we will add some animation to make the orb spin. This time we will use the rotation property to make the sphere spin on its axis:

```
property: rotation;
from: 0 0 0;
to: 00 -360 0;
dur: 3000;
loop: true;
dir: normal;
 startEvents: click;
        "
```

Test your sphere and make sure it works…

You should get a nice spinning motion that makes it look a little more magical!

7.2 Adding our Dragon

Next, we will add the centrepiece of our Dragon Realm!

To start we will define an asset and then nest inside it an asset item, just like we did for the shark. Place the following under the dragon comment in your code:


```
<a-assets>
      <a-asset-item>

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

Here are the properties to add to the asset-item tag:

- `id="dragon"`
- `src="https://your_link_here">`

Remember: You will need to generate a URL for your dragon and add it to the property above. Follow the steps from the guide part 2, in section **6.4 Uploading your own GLFT files**

You can get the dragon file to upload to Glitch.com here:

http://www.ai3.academy/inspire/metaverse/Asian-style_dragon.glb

This only sets up our dragon model as an asset. Next we need to add it to our project….

Let's set the dragon up as an entity. You can add this under the asset tags you just created:

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

And now add the properties:

```
id="spindragon"
gltf-model="#dragon"
position="0 0 0"
scale="5 5 5"
```

#dragon will look up the ID that we set on the asset earlier.

We will use the new id 'spindragon' later….

Now run your code and see what happens?

No Dragon sighting yet?

That's because it's hiding inside the magic orb, or could that be a dragon egg?

Let's animate it to get it out!

This is a bit of a more complex animation as we want our dragon to do ***four things*** at once:

1) We want to trigger it coming out the orb by looking at the orb
2) We want the dragon to move upwards out the orb
3) We want the dragon to grow (scale) as it rises up
4) And lets make it spin… just because it looks cool!

First we will focus on the animations, and we will add the animation trigger last.

Animating the dragon

In the properties of the entity tag we created on the previous page, add the following code:

```
animation__1="
    property: rotation;
    from: 0 0 0;
    to: 00 -360 0;
    dur: 3000;
    loop: true;
    startEvents: dragongrow;
    "

    animation__2="
    property: position;
    from: 0 0 0;
    to: 0 2 0;
    dur: 6000;
    startEvents: dragongrow;
    "

    animation__3="
    property: scale;
    from: 6 6 6;
    to: 70 70 70;
    dur: 8000;
    startEvents: dragongrow;
    "
```

Notice how our code layout now makes looking at three animations in the same property much easier?

Each separate animation needs to have a number in sequential order. Note that before each number are **TWO** underscore lines like these _

This is the event we will add in a moment. The animation will wait for this trigger before starting. We will add this trigger to the orb now….

Note: If you would like to test your code before continuing to see if your dragon is there, you could change the Y position of your dragon by adding 1 to it. It should be sitting just above the orb at this position.

Don't forget to change it back!

Triggering the event

The last thing we need to do is ***trigger the event*** of the dragon animation.

As we have set a global fuse on our project, we can do this by looking at the orb.

Because we want to trigger one animation by interacting with another, we need to add some special code.

The first thing we need to do is make sure our code is in the right sequence, as this is very important for it to work. To do this we need to understand PARENT and CHILDREN code.

You will remember we started with a sphere tag for our orb:

```
<a-sphere> </a-sphere>
```

In our code this is called the PARENT, because it's the top level of our process.
The CHILD(REN) will be the entity tag where we created our dragon.

Here is the structure our code needs to have:

```
<a-assets>
      <a-asset-item>
            Dragon model id and properties
      </a-asset-item>
</a-assets>

<a-sphere>
      Sphere animation properties are here
      <a-entity>
            Dragon animation properties are here
      </a-entity>
</a-sphere>
```

Next, we need to add an extra property to our sphere:

- `proxy-event__enter="event: mouseenter; to: CHILDREN; as: dragongrow"`

This line tells the orb on a mouse enter (our trigger by looking at the orb) to run the 'dragongrow' events in the CHILDREN, or lower level code, which is our dragon entity in this case.

Now there is just one more addition to get our event to run.

As this event command isn't in the standard JavaScript library, we need to add a new library file to the in the <Head> tag at the top of our code. Add this line under the scripts that are already there:

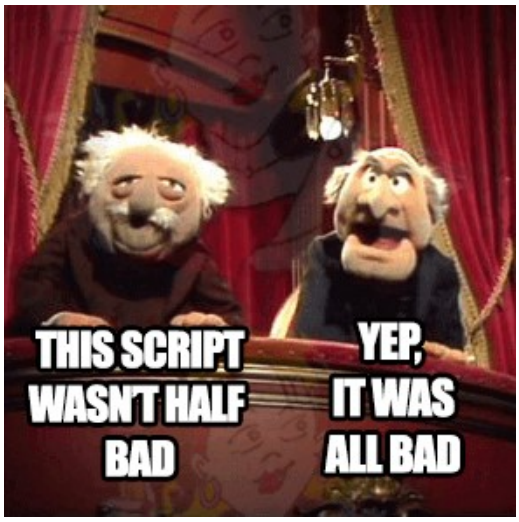
```
  <script src="https://unpkg.com/aframe-proxy-event-component/dist/aframe-
proxy-event-component.min.js"></script>
```

7.3 Catching a dragon

Now we should have a pretty cool animation of a dragon spinning and growing as it rises above the orb.

If you like you could change the dragon into a fairy, power up, or anything that could work within your story.

We can also turn this into a fun game to play in your VR world by running a variable that counts how many dragons we hatch.



JavaScript and A-Frame – a perfect match!

Toward the top of your HTML template, you will see a comment called *'JavaScript coding section'*

Find the `<script> </script>` tags under it and add the following JavaScript code:

```
var count = 0;
function jsfunc(){
    count += 1;
    console.log("Dragon Hatched");
     console.log(count);
}
```

Creating a variable called count, and setting it's start value to 0

Create a function called ***jsfunc***

Add one to our variable count

Display our variable count in the console log

Display 'Dragon Hatched' in the console log

Now we just need to make one last property addition to our sphere to tell it to run our JavaScript function as well as launching our dragon when we look at it:

- `onclick="jsfunc()"`

Accessing the console log

The console log is a powerful testing tool that coders often use to check how parts of their code is functioning.

Here is how to open the console log in all the popular browsers:

https://appuals.com/open-browser-console/

- Try running your code and then checking the console log after you hatch your dragon!

JavaScript Challenge!

Can you research how to display your variable on the screen rather than the console log?

For example, you could have a ***Dragons Hatched:*** count on your screen and add multiple orbs to find in your world.

More Ideas….

You could add the same system for collecting food, materials or gold and turn your world into an adventure game!

7.4 Background Music

Adding music and sound effects can really add another element to your users VR experience. Here is a great little site that has a simple way to add background music:

https://joshondesign.com/2017/07/24/aframe_tips

END OF PART 3

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

Item	Syntax
