First, the viewport is the entire area on your browser.

The viewport is the content you can see.

Instead of using 800 and 600 in sizes, we can use window.innerWidth

const sizes = {

  width: window.innerWidth,

  height: window.innerHeight,

};

This will fullscreen the canvas pretty much

One issue is we have a margin on top and bottom and around screen. This is a matter of styling.

This is because we have a margin in the <body> for the css.

body {

  margin: 0;

}

\* {

  margin: 0;

  padding: 0;

}

Can use either solution.

Only need \* though.

By adding this we can get rid of the scroll bar:

.webgl {

  position: fixed;

  top: 0;

}

Even better for the screen. Now we want to get rid of the blue outline.

So we want to add:

.webgl {

  position: fixed;

  top: 0;

  left: 0;

  outline: none;

}

We also need to disable the scrolling trackpad of endless scroll similar to scrollview in react native. This can happen on some OS.

Also, if it is just an animation and you do not need controls, this can happen when people try to scroll showing white space at the bottom.

html,

body {

  overflow: hidden;

}

So we add this to the css to not allow overflow.

The problem we get now is that if we resize the window, it will not change the screen size and just show white space. So we need to be aware if a resize occurred.

So we add an event listener to the window resize and we need to then update our object and our camera with the new size and aspect ratio respectively

window.addEventListener("resize", (event) => {

  //update sizes

  sizes.wdith = window.innerWidth;

  sized.height = window.innerHeight;

  //update camera aspect

  camera.aspect = sizes.width / sizes.height;

});

Now we need to update the camera object

Although now we need to also update our renderer

 renderer.setSize(sizes.width, sizes.height);

so on event, update sizes, camera, and render

Now we need to talk about pixel ratio. You might see some blur or a stairs effect on edges because you have a pixel ratio greater than one.

The pixel ratio is how many pixels you have based on how many you need for the image or thing on the screen. Apple started building screens with pixel ratio 2 or more.

Some build beyond 3.

A pixel ratio of 2 means 4 times more pixels to render. 3 means 9 and so on. The highest pixel ratios are often on the weakest devices. Basically mobile phones.

Very hard for your gpu to render this many pixels.

Pixel ratio of 2 is pretty much the limit where you can tell a difference.

Check your device pixel ratio in your browser console.

Type window.devicePixelRatio into your console.

I have 1.25 lol wtf

Now we can set our renderer pixel ratio

renderer.setPixelRatio(window.devicePixelRatio);

but we need to limit the value to 2 incase a device is too strong or too many pixel ratios.

renderer.setPixelRatio(Math.min(window.devicePixelRatio, 2));

we also want to add this code into the resize listener incase someone moves a browser to a separate screen.

renderer.setPixelRatio(Math.min(window.devicePixelRatio, 2));

Now we want to handle full screen similar to a game that will take over computer.

In this example we will listen to a double click.

First we want to test and see if user wants to try and leave full screen.

window.addEventListener("dblclick", () => {

  console.log("double click");

});

In order to go full screen, we have to use requestFullScreen()

We must do this on an element, but you can do this with any element.

window.addEventListener("dblclick", () => {

  if (!document.fullscreenElement) {

    canvas.requestFullscreen();

    // console.log("go fullscreen");

  } else {

    document.exitFullscreen();

  }

});

Here is what it looks like and now on double click the canvas will enter and exit fullscreen.

Unfortunately, this does not work on Safari.

To make this work on Safari, we need to use document web kit or something.

window.addEventListener("dblclick", () => {

  const fullscreenElement =

    document.fullscreenElement || document.webkitFullscreenElement;

  if (!fullscreenElement) {

    if (canvas.requestFullscreen) {

      canvas.requestFullscreen();

    } else if (canvas.webkitRequestFullScreen) {

      canvas.webkitRequestFullscreen();

    }

  } else {

    if (document.exitFullscreen) {

      document.exitFullscreen();

    } else if (document.webkitExitFullscreen) {

      document.webkitExitFullscreen();

    }

  }

});

Here is what code looks like to get the fullscreen thing to work on any browser pretty much. Given safari is obsolete, the original code seemed to work on edge as well. I would just have to test FireFox.