

Coding 1 – Homework.

Please find below the links for all Homework Mimic project.

Bellow you can find the link for the GitHub Repository (the Zip file was too big for the submission).

<https://github.com/22046361/Coding--1-Homework-Submission.git>

On GitHub you can also find the screen recordings for each week's homework.

Therefore, for each week you can find:

- A file with the link to the Mimic project.
- A separate file with the code.
- A screen-recording of the weekly project.

- **Links for Mimic project:**

Week 1: <https://mimicproject.com/code/c80ab1a1-421d-4c7f-6461-aa905bf0f1fa>

Week 2: <https://mimicproject.com/code/bf2c6d8e-d3cf-8de9-bab0-66231629a4a6>

Week 3: <https://mimicproject.com/code/fc583ec4-dfa7-b513-5b05-2cb04fb7de64>

Week 4: <https://mimicproject.com/code/6f045448-8f5d-97f3-cfbf-4393b978f77e>

Week 5 - Recap.

Week 6: <https://mimicproject.com/code/945e69e0-197d-d8ae-7790-e691334a321a>

Week 7: <https://mimicproject.com/code/a3758580-3e5e-fefa-6863-89579a259dc1>

Week 8: Homework 1A <https://mimicproject.com/code/d1bbb203-5975-4479-5cd4-dd4145611dde>

Homework 1B <https://mimicproject.com/code/d10e362c-270c-bc8b-1a75-e7e761fde89d>

Homework 2A & B <https://mimicproject.com/code/695bfe9f-9d93-347e-e81d-35037e861b1f>

Homework 2C <https://mimicproject.com/code/976e929e-e2a9-c63f-4396-fbf680bf0b7c>

- Codes for each week's homework projects:

## **Week 1.**

The screenshot shows a browser window with two tabs: "maximilian.js" and "HomeWork Week 1 by kzaromytidou". The "maximilian.js" tab contains the following code:

```
// HOMEWORK WEEK 1 - GETTING STARTED SEED
// THANK YOU FOR PROVIDING THE CODE TO START OFF
MANY THANKS TO MAXIMILIAN.JS FOR THE CODE

STOP
```

The "HomeWork Week 1 by kzaromytidou" tab displays the rendered HTML from the code. It includes a title "Homework Week 1 - Getting started Seed", a "thank you" message, and a "play" button. The code editor on the right shows the full HTML and JavaScript code for the page.

```
> 4   <script src = "https://mimicproject.com/libs/nexusUI.js">
</script>
5   <script src = "https://mimicproject.com/libs/maximilian.v.0.1.js">
</script>
6   <link href="https://fonts.googleapis.com/css?family=Staatliches" rel="stylesheet">
7   <link rel="stylesheet" href="https://mimicproject.com/libs/maximilian-example.css" />
8   <div>
9     <div id="title"> Homework Week 1 - Getting started
10    Seed</div>
11
12    // thank you for providing the code to start off
13    <div> Many thanks to MAXIMILIAN.JS for the code </div>
14    <div><button id="playButton">Play</button></div>
15  </div>
16  <div id="oscilloscope"></div>
17  <div id="spectrogram"></div>
18 </head>
19 <body>
20
21  <!-- Maximilian code goes here -->
22  <script id = "myAudioScript">
23    var osc1 = new Maximilian.maxiOsc();
24    var osc2 = new Maximilian.maxiOsc();
25
26    // Adding 2 more oscillators.
27    // Osc3 & Osc4 are part of the homework requirements, while osc5 is added as a frequency modulator to the wave.
28    var osc3 = new Maximilian.maxiOsc();
29    var osc4 = new Maximilian.maxiOsc();
30    var osc5 = new Maximilian.maxiOsc();
31
32
33    // trying to keep the frequency low through the modulator.
34    function play() {
35      var freq = (Math.sin((osc1.freq * 100) + osc2.freq * 10.1) + osc3.freq * 0.01) * 1000000;
```

```
<!DOCTYPE html>
<html>
  <head>
    <script src = "https://mimicproject.com/libs/nexusUI.js"></script>
    <script src = "https://mimicproject.com/libs/maximilian.v.0.1.js"></script>
    <link href="https://fonts.googleapis.com/css?family=Staatliches" rel="stylesheet"/>
    <link rel="stylesheet" href="https://mimicproject.com/libs/maximilian-example.css" />
  <div>
    <div id="title"> Homework Week 1 - Getting started
      Seed</div>
```

```

// thank you for providing the code to start off
<div> Many thanks to MAXIMILIAN.JS for the code </div>
<div><button id="playButton">Play</button></div>
</div>
<div id="oscilloscope"></div>
<div id="spectrogram"></div>
</head>
<body>

<!-- Maximilian code goes here -->
<script id = "myAudioScript">
    var osc1 = new Maximilian.maxiOsc();
    var osc2 = new Maximilian.maxiOsc();

    // Adding 2 more osclators.
    // Osc3 & Osc4 are part ofthe homework requirements, while osc5 is added as a frequency
    modulator to the wave.
    var osc3 = new Maximilian.maxiOsc();
    var osc4 = new Maximilian.maxiOsc();
    var osc5 = new Maximilian.maxiOsc();

    // trying to keep the frequency low throug the modulator.
    function play() {
        return (osc1.saw(10) * osc2.saw(10.1) + osc3.saw(20) * osc4.saw(20.1)) * osc5.saw(0.06)

    }

    // tried also to play a bit... chekc this out, but lower volume before

    // function play() {
    // return ((osc1.saw(10) * osc2.saw(10.1)) / (osc3.saw(20) * osc4.saw(20.1))) * osc5.saw(0.06)

    // }

</script>

<!-- Main Javascript code goes here -->
<script language="javascript">
    let maxi;
    initAudioEngine().then((dspEngine)=>{
        maxi = dspEngine;
        setup();
        //Get audio code from script element

```

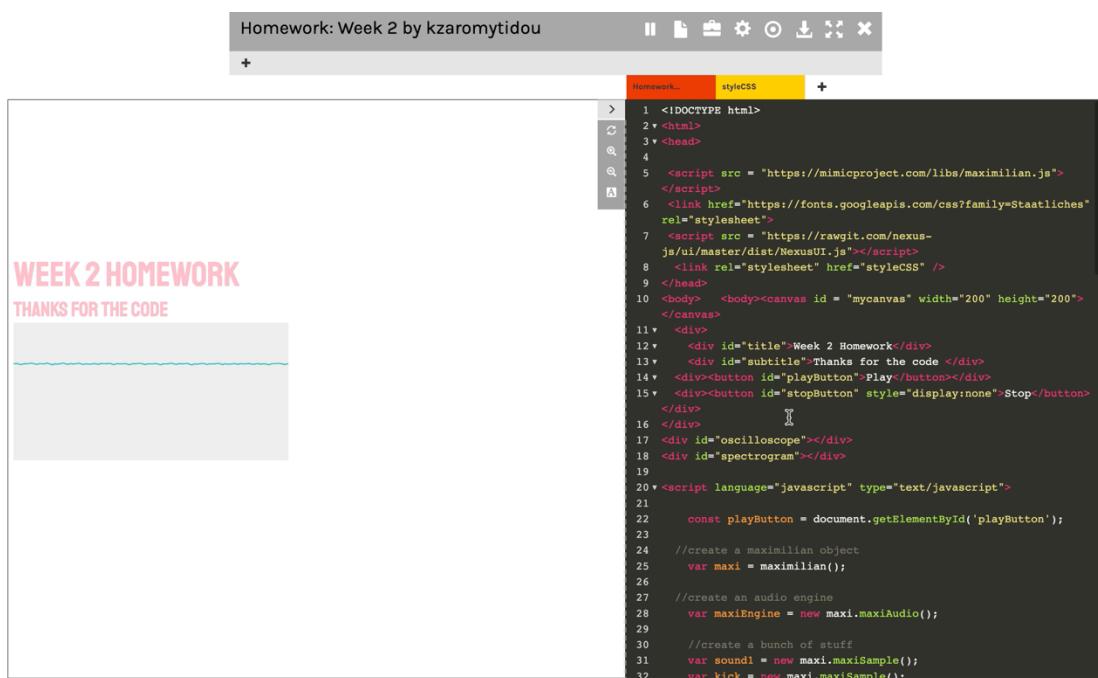
```

maxi.setAudioCode("myAudioScript");
}

/////////YOU CAN IGNORE ME - CODE FOR SCOPES/////////
let setup = ()=> {
  maxi.hush()
  Nexus.context = maxi.audioWorkletNode.context;
  new Nexus.Oscilloscope('oscilloscope', {'size': [400,100]}).connect(maxi.audioWorkletNode);
  new Nexus.Spectrogram('spectrogram', {'size': [400,100]}).connect(maxi.audioWorkletNode);
  const playButton = document.getElementById('playButton');
  let playAudio = () => {
    playButton.innerHTML = maxi.play() ? "STOP":"PLAY"
  }
  playButton.addEventListener("click", () => playAudio());
}
///////////////////////////////
</script>
</body>
</html>

```

## Week 2



The screenshot shows a browser window with the title "Homework: Week 2 by kzaromytidou". Inside the browser, there is a simple web page with the following content:

**WEEK 2 HOMEWORK**  
THANKS FOR THE CODE

The browser's developer tools are open, specifically the Elements tab. The code visible in the Elements tab is:

```

1  <!DOCTYPE html>
2  <html>
3  <head>
4
5  <script src = "https://mimicproject.com/libs/maximilian.js">
6  </script>
7  <link href="https://fonts.googleapis.com/css?family=Staatliches" rel="stylesheet">
8  <script src = "https://rawgit.com/nexusjs/ui/master/dist/NexusUI.js"></script>
9  <link rel="stylesheet" href="styleCSS" />
10 <head>
11 <body> <body><canvas id = "mycanvas" width="200" height="200">
12 </div>
13 <div id="title">Week 2 Homework</div>
14 <div id="subtitle">Thanks for the code</div>
15 <div><button id="playButton">Play</button></div>
16 <div><button id="stopButton" style="display:none">Stop</button>
17 </div>
18 <div id="oscilloscope"></div>
19 <div id="spectrogram"></div>
20 <script language="javascript" type="text/javascript">
21
22   const playButton = document.getElementById('playButton');
23
24   //create a maximilian object
25   var maxi = maximilian();
26
27   //create an audio engine
28   var maxiEngine = new maxi.maxiAudio();
29
30   //create a bunch of stuff
31   var sound1 = new maxi.maxiSample();
32   var kick = new maxi.maxiSample();
33
34 </script>

```

```

<!DOCTYPE
html>

<html>
<head>

<script src = "https://mimicproject.com/libs/maximilian.js"></script>
<link href="https://fonts.googleapis.com/css?family=Staatliches" rel="stylesheet">
<script src = "https://rawgit.com/nexus-js/ui/master/dist/NexusUI.js"></script>
<link rel="stylesheet" href="styleCSS" />
</head>
<body> <body><canvas id = "mycanvas" width="200" height="200"> </canvas>
<div>
<div id="title">Week 2 Homework</div>
<div id="subtitle">Thanks for the code </div>
<div><button id="playButton">Play</button></div>
<div><button id="stopButton" style="display:none">Stop</button></div>
</div>
<div id="oscilloscope"></div>
<div id="spectrogram"></div>

<script language="javascript" type="text/javascript">

    const playButton = document.getElementById('playButton');

    //create a maximilian object
    var maxi = maximilian();

    //create an audio engine
    var maxiEngine = new maxi.maxiAudio();

        //create a bunch of stuff
    var sound1 = new maxi.maxiSample();
    var kick = new maxi.maxiSample();
    var basis = new maxi.maxiSample();
    var myOsc = new maxi.maxiOsc();
    var myOsc2 = new maxi.maxiOsc();
    var myClock = new maxi.maxiClock();

    //var kickCount = 9;

        var scratch=0;
    var counter = 0;
    //var myTempo = 200;
    // var delayTime = myTempo * 3;
    myClock.setTempo(70);
    myClock.setTicksPerBeat(2);

```

```

var oscilloscope,spectrogram;

let playAudio = () => {
    playButton.style.display="none";
    //initiating the audio engine.
    maxiEngine.init();
    //the samples are from https://freesound.org/people/schafferdavid/sounds/211471/
    maxiEngine.loadSample("258021__soundscape-humfak_leaves.wav", sound1);
    maxiEngine.loadSample("659627__josefpres_piano-loops-036-octave-down-long-loop-120-
bpm.wav", kick);
    maxiEngine.loadSample("659628__garuda1982_acoustic-guitar-melody-fingerstyle-9.mp3",
basis);

    //show an oscilloscope and freqscope - copied this entirely from existing code.
Nexus.context = maxiEngine.context;
    oscilloscope = new Nexus.Oscilloscope('oscilloscope', {'size':
[400,100]}).connect(maxiEngine.maxiAudioProcessor);
    spectrogram = new Nexus.Spectrogram('spectrogram', {'size':
[400,100]}).connect(maxiEngine.maxiAudioProcessor);

    maxiEngine.play = function() {

myClock.ticker();

if (myClock.tick) {

    scratch=0;
    counter++;
}

if (myClock.tick && counter%8==2) {
    kick.trigger();
}
if (myClock.tick && counter%16==2) {
    kick.trigger();
}

if (myClock.tick && counter%8==4) {

sound1.trigger();

}

```

```

        if (myClock.tick && counter%16==7) {
            basis.trigger();
        }
        // played with the numbers to abjust the rythem.
        var out = kick.playOnce() + sound1.playOnce() + basis.play(2) + myOsc.sinebuf(2);
        return out * 0.2;
        // return (out + myDelay.dl(out,delayTime,feedback)) * 0.0;

    }

}

playButton.addEventListener("click", () => playAudio());


</script>
</body>
</html>

```

### CSS STyles

```

#inner-container {
    background-color: #40FFFF;
    width:900px;
    padding:100px;
    position:absolute;
}

#main-container {
    background-color: #40ffff;
    width:auto;
    height:100px;
}

body {
    font-family: 'Staatliches';
    font-size: 50px;
    color: pink;
    text-align: left;
}

```

```

}

#subtitle {
    font-size: 60%;

}

button {
    background-color: pink;
    color: black;
    border:solid black 4px;
    width:150px;
    padding:10px;
    margin:10px;
    font-family: 'Staatliches';
    font-size: 25px;
}

button:hover {
    background-color: transparent;
    color:#FFff10;
    border:solid #FFD12C 5px;
}

```

## Week 3

New HW Week3 by kzaromytidou

Homework Week 3

```

New HW W... + New HW W...
+ + New HW W...
> 1 <!DOCTYPE html>
<html>
<body style="background-color:black;">
<h1 id="this-header"> Homework Week 3 </h1>
<canvas></canvas>
<script type="text/javascript">
let header1 = document.getElementById("this-header");
header1.style.color = "yellow";
// This is where we are going to store the mouse information
var mousex = 0;
var mousey = 0;
// We really need this
var TWO_PI = Math.PI * 2;
// This gets a reference to the canvas in the browser
var canvas = document.querySelector("canvas");
// This sets the width and height to the document window
canvas.width = window.innerWidth;
canvas.height = window.innerHeight;
// Be aware that when you resize the window, you will need to
call (do) this again
// Creating the context for the canvas.
var context = canvas.getContext("2d");
// adding the rotate function to manipulate the shape.
context.rotate(45 * Math.PI / 180);
//making the mouse interactive.
canvas.addEventListener('mousemove', getMouse, false);

```

```
<!DOCTYPE
html>
<html>

<body style="background-color:black;">

<h1 id="this-header"> Homework Week 3 </h1>

<canvas></canvas>

<script type="text/javascript">
let header1 = document.getElementById("this-header");
header1.style.color = "yellow";
// This is where we are going to store the mouse information
var mouseX = 0;
var mouseY = 0;
// We really need this
var TWO_PI = Math.PI * 2;

// This gets a reference to the canvas in the browser

var canvas = document.querySelector("canvas");

// This sets the width and height to the document window
canvas.width = window.innerWidth;
canvas.height = window.innerHeight;
// Be aware that when you resize the window, you will need to call (do) this again

// Creating the context for the canvas.
var context = canvas.getContext("2d");

// adding the rotate function to manipulate the shape.

context.rotate(45 * Math.PI / 180);
//making the mouse interactive.
canvas.addEventListener('mousemove', getMouse, false);

function getMouse(mousePosition) {
    mouseX = mousePosition.layerX;
    mouseY = mousePosition.layerY;
}
```

```

// This function translates the canvas so that we're looking at it from a different position, meaning
that 0,0 is somewhere else
function draw_one() {

    var segments = 200;
    var spacing = TWO_PI / segments;
    var radius = 200;

    //clear the screen
    context.clearRect(0,0, canvas.width, canvas.height);

    //Drawing first shape.

    context.beginPath();

    for (var i = 0; i < segments; i++) {

        context.strokeStyle = "FF3010"; //set the line colour to black
        var x = 550 + Math.cos(spacing / 2 * i * (mouseX / 50)) * Math.cos(spacing * i * (mouseY / 50)) *
radius;
        var y = Math.sin(spacing / 2 * i * (mouseX / 50)) * Math.sin(spacing * i * (mouseY / 50)) * radius;

        context.lineTo(x + radius, y + radius);

    }

    context.stroke(); //draw the outline
    context.closePath();
    requestAnimationFrame(draw_one);

}

//Drawing second shape.
function draw_two() {

    var segments1 = 1000;
    var spacing1 = TWO_PI / segments1;
    var radius1 = 400;

    context.beginPath();

    for (var i1 = 0; i1 < segments1; i1++) {

        context.strokeStyle = "#FF5090"; //set the line colour to black

```

```

        var x1 = Math.cos(spacing1 * i1 * (mouseX / 50)) * Math.cos(spacing1 * i1 * (mouseY / 50)) *
radius1;
        var y1 = Math.sin(spacing1 * i1 * (mouseX / 50)) * Math.sin(spacing1 * i1 * (mouseY / 50)) *
radius1;

        context.lineTo(x1 + radius1 + 100, y1 + radius1 + 50);

    }

    context.stroke(); //draw the outline
    context.closePath();
    requestAnimationFrame(draw_two);

}

//Drawing third shape.
function draw_three() {

    var segments2 = 1000;
    var spacing2 = TWO_PI / segments2;
    var radius2 = 750;

    context.beginPath();

    for (var i2 = 0; i2 < segments2; i2++) {

        context.strokeStyle = "#FFFFFF"; //set the line colour to black
        var x2 = Math.cos(spacing2 * i2 * (mouseX / 50)) * Math.cos(spacing2 * i2 * (mouseY / 150)) *
radius2*2;
        var y2 = Math.sin(spacing2 * i2 * (mouseX / 50)) * Math.sin(spacing2 * i2 * (mouseY / 50)) *
radius2;

        context.lineTo(x2 + radius2, y2 + radius2);

    }

    context.stroke(); //draw the outline
    context.closePath();
    requestAnimationFrame(draw_three);

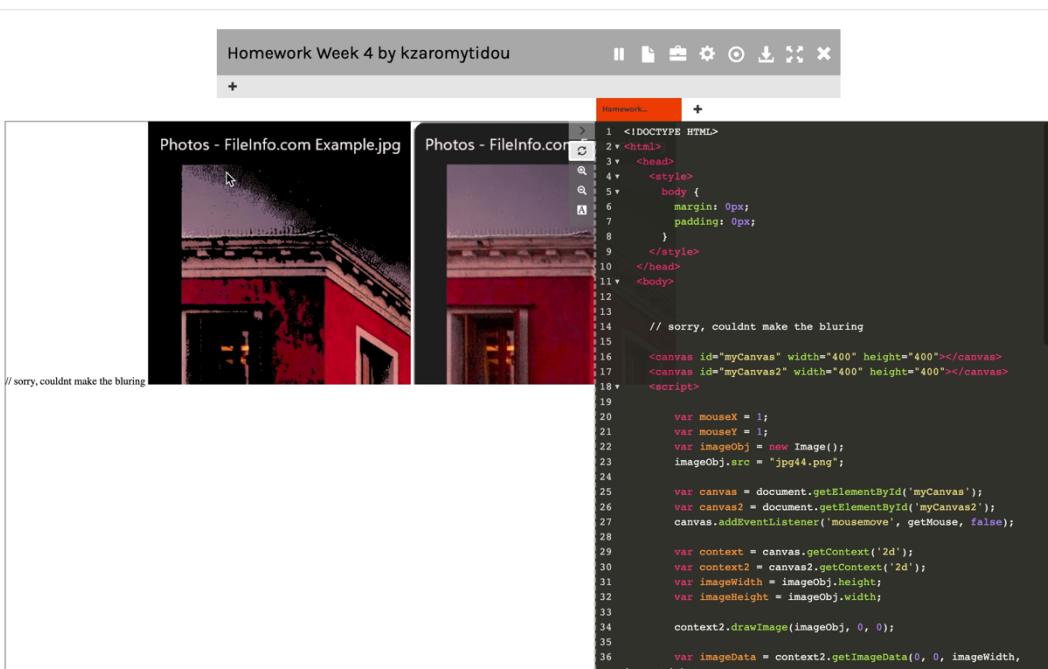
}

// run the seperate draw functions
requestAnimationFrame(draw_one);
requestAnimationFrame(draw_two);
requestAnimationFrame(draw_three);

```

```
</script>  
</body>  
  
</html>
```

## Week 4.



```
<!DOCTYPE  
HTML>  
  
<html>  
<head>  
<style>  
body {  
    margin: 0px;  
    padding: 0px;  
}  
</style>  
</head>  
<body>  
  
// couldnt make the bluring  
  
<canvas id="myCanvas" width="400" height="400"></canvas>
```

```

<canvas id="myCanvas2" width="400" height="400"></canvas>
<script>

    var mouseX = 1;
    var mouseY = 1;
    var imageObj = new Image();
    imageObj.src = "jpg44.png";

    var canvas = document.getElementById('myCanvas');
    var canvas2 = document.getElementById('myCanvas2');
    canvas.addEventListener('mousemove', getMouse, false);

    var context = canvas.getContext('2d');
    var context2 = canvas2.getContext('2d');
    var imageWidth = imageObj.height;
    var imageHeight = imageObj.width;

    context2.drawImage(imageObj, 0, 0);

    var imageData = context2.getImageData(0, 0, imageWidth, imageHeight);
    var data = imageData.data;
    var imageData2 = context.getImageData(0, 0, imageWidth, imageHeight);
    //var imageData2 = imageData;

    var draw = function() {

        // iterate over all pixels
        for(var i = 0; i < imageHeight; i++) {
            // loop through each row
            for(var j = 0; j < imageWidth; j++) {

                if (data[((imageWidth * i) + j) * 4] > mouseX) {

                    imageData2.data[((imageWidth * i) + j) * 4] = data[((imageWidth * i) + j) * 4];
                    imageData2.data[((imageWidth * i) + j) * 4+1] = data[((imageWidth * i) + j) * 4 + 1];
                    imageData2.data[((imageWidth * i) + j) * 4+2] = data[((imageWidth * i) + j) * 4 + 2];
                    imageData2.data[((imageWidth * i) + j) * 4+3] = data[((imageWidth * i) + j) * 4 + 3];

                }

            }
        }

        else {

            imageData2.data[((imageWidth * i) + j) * 4] = 0;
            imageData2.data[((imageWidth * i) + j) * 4+1] = 0;
            imageData2.data[((imageWidth * i) + j) * 4+2] = 0;
            imageData2.data[((imageWidth * i) + j) * 4+3] = 255;

        }

    }

}


```

```

        }
    }
}

context.putImageData(imageData2,0,0);
requestAnimationFrame(draw);
};

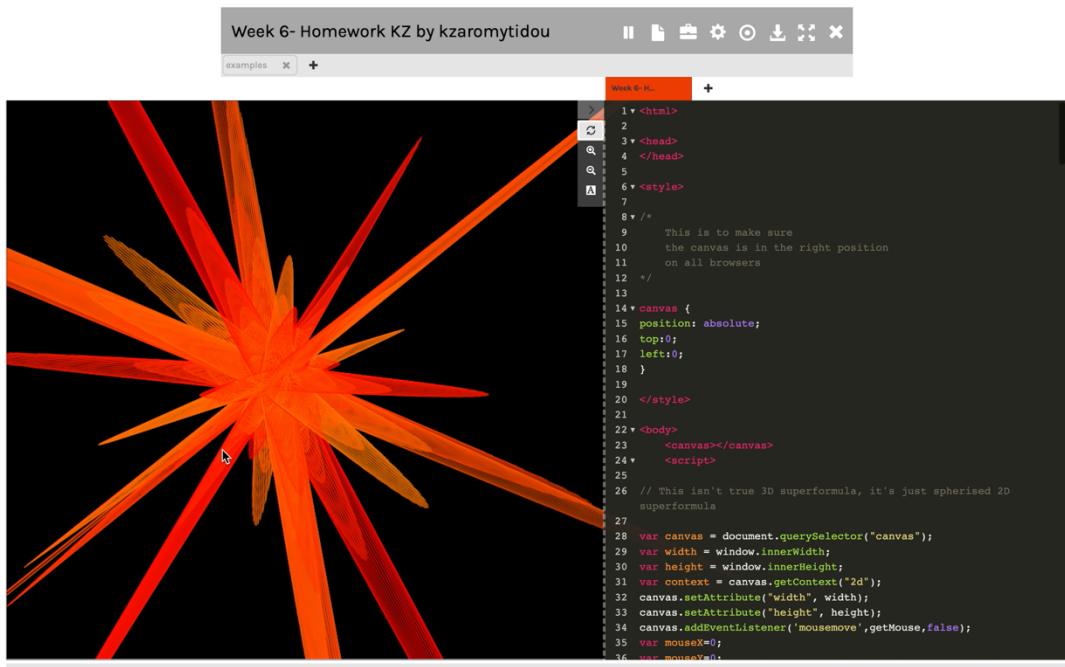
requestAnimationFrame(draw);

function getMouse(mousePosition) {
    mouseX = mousePosition.layerX;
    mouseY = mousePosition.layerY;
}

</script>
</body>
</html>

```

## Week 6.



```
<html>
>

    <head>
    </head>

    <style>

        /*
        This is to make sure
        the canvas is in the right position
        on all browsers
        */

        canvas {
            position: absolute;
            top:0;
            left:0;
        }

    </style>

    <body>
        <canvas></canvas>
        <script>

            // This isn't true 3D superformula, it's just spherised 2D superformula

            var canvas = document.querySelector("canvas");
            var width = window.innerWidth;
            var height = window.innerHeight;
            var context = canvas.getContext("2d");
            canvas.setAttribute("width", width);
            canvas.setAttribute("height", height);
            canvas.addEventListener('mousemove', getMouse, false);
            var mouseX=0;
            var mouseY=0;

            var fov = 500;

            var point = [];
            var point3d = [];
            var angleX = 0;
            var angleY = 0;
            var HALF_WIDTH = width / 2;
            var HALF_HEIGHT = height / 2;
```

```

var x3d = 0;
var y3d = 0;
var z3d = 0;

var firstx2d=0;
var firstly2d=0;
var firstScale=0;
var lastScale = 0;
var lastx2d = 0;
var lasty2d = 0;

var elements = 300;
var x, y = 0;
var lastX, lastY = 0;
var firstX, firstY = 0;
var r1, r2 = 0;
// var spacing = (Math.PI * 2) / elements;
var m = 0;
var n1 = 0;
var n2 = 0;
var n3 = 0;
var b = 1;
var a = 1;

// The below code creates a sphere of points
var dim = 120; // This is the number of rings
// Each ring has as many points as there are rings
// This is the spacing for each ring
var spacing = ((Math.PI * 9 +3) / dim);

var numPoints = dim * dim; // This is the total number of points

var size = 5; // This is the size.
var counter=0;

function draw() {

var mouseX1=mouseX/50;
var mouseY1=mouseY/50;
//var mouseY1=1;

m = Math.floor((mouseY / height) * 15);

```

```

n1 = (mouseX / width) - 10;

//change these to different things for different shapes.
n2 = n3 = n1;

// We're doing the geometry in the draw loop because we want to interact with it.

var points = [];
// Now we build the geom

// This is a sphere just like before
for (var i = 0; i < dim; i++) {

    //Use r to calculate x and y
    var z = size * Math.cos(spacing / 2 * i) * (dim/8);

    // Calculate the size of the current ring
    var s = size * Math.sin(spacing / 2 * i);

    // For each ring

    for (var j = 0; j < dim; j++) {

        r1 = size * Math.pow(Math.pow(Math.abs(Math.cos((m * spacing * j) / 4) / a), n2) +
        Math.pow(Math.abs(Math.sin((m * spacing * j) / 4)) / b, n3), -(1 / n1));
        // Create a circle at the current size, at the current depth

        var point = [r1 * Math.cos(spacing * j) * s,r1 * Math.sin(spacing * j)*s,z];

        // Add the points
        points.push(point);

    }
}

context.fillStyle = "rgb(0,0,0)";
context.fillRect(0, 0, width, height);

// angleX+=((mouseX/width)-0.5)/4;
// angleY+=((mouseY/height)-0.5)/4;

//angleX+=0.01;
//angleY+=0.01;
angleX+=((mouseX/width))/20;
angleY+=((mouseY/height))/20;

```

```

// Here we run through each ring and work out where it should be drawn

for (let i = 0; i < numPoints; i+=dim) {

    for (let j = 0; j < dim; j++ ) {
        point3d = points[Math.floor(i+j)];
        z3d = point3d[2];

        // This is the speed of the z
        // It moves the points forwards in space
        // We don't need it for the pure rotate
        // z3d -= 1.0;

        // Check that the points aren't disappearing into space and if so push them back
        // This also stops them stretching
        // When they get too close
        if (z3d < -fov) z3d += 0;

        point3d[2] = z3d;

        // Calculate the rotation

        rotateX(point3d,angleX);
        rotateY(point3d,angleY);

        // Get the point in position

        x3d = point3d[0];
        y3d = point3d[1];
        z3d = point3d[2];
        // Convert the Z value to a scale factor
        // This will give the appearance of depth
        var scale = (fov / (fov + z3d));

        // Store the X value with the scaling
        // FOV is taken into account
        // (just pushing it over to the left a bit too)
        var x2d = (x3d * scale) + HALF_WIDTH / 2;

        // Store the Y value with the scaling
        // FOV is taken into account

        var y2d = (y3d * scale) + HALF_HEIGHT;

        // // If our main loop is going to join all the points together in a line, we need to store the first points and use
        them at the end.

```

```

if (j==0){

    firstx2d=x2d;
    firsty2d=y2d;
    firstScale=scale;

    lastx2d=x2d;
    lasty2d=y2d;
    lastScale=scale;

}

// Draw the point

// Set the size based on scaling
context.lineWidth = scale;

context.strokeStyle = "rgba(" + i + "," + j + "," + 0 + "," + scale/2 + ")";

// context.strokeStyle = "rgb(100,65,85)";
context.beginPath();
context.moveTo(lastx2d + lastScale, lasty2d);
context.lineTo(x2d + scale, y2d);
context.stroke();

// Store the last point so we can join it to the next one.

lastx2d=x2d;
lasty2d=y2d;
lastScale=scale;

// if it's the end of the current ring, join it to the first

if (j==dim-1 {

    context.lineWidth = scale;

//context.strokeStyle = "rgb(255,255,255)";

var colourB = (mouseY-100)/2;

```

```

        context.beginPath();
        context.moveTo(lastx2d + lastScale, lasty2d);
        context.lineTo(firstx2d + firstScale, firsty2d);
        context.stroke();

    }

}

}

setInterval(draw, 30);

function rotateX(point3d,angleX) {
    var x = point3d[0];
    var z = point3d[2];

    var cosRY = Math.cos(angleX);
    var sinRY = Math.sin(angleX);

    var tempz = z;
    var tempx = x;

    x= (tempx*cosRY)+(tempz*sinRY);
    z= (tempx*-sinRY)+(tempz*cosRY);

    point3d[0] = x;
    point3d[2] = z;

}

function rotateY(point3d,angleY) {
    var y = point3d[1];
    var z = point3d[2];

    var cosRX = Math.cos(angleY);
    var sinRX = Math.sin(angleY);

    var tempz = z;
    var tempy = y;

    y= (tempy*cosRX)+(tempz*sinRX);
    z= (tempy*-sinRX)+(tempz*cosRX);
}

```

```
    point3d[1] = y;
    point3d[2] = z;

}

//here's our function 'getMouse'.
function getMouse (mousePosition) {
//for other browsers..
// mouseX = mousePosition.layerX +10 ;
// mouseY = mousePosition.layerY;
if (mousePosition.layerX || mousePosition.layerX === 0) {
mouseX = mousePosition.layerX;
mouseY = mousePosition.layerY;
} else if (mousePosition.offsetX || mousePosition.offsetX === 0) {
mouseX = mousePosition.offsetX;
mouseY = mousePosition.offsetY;
}

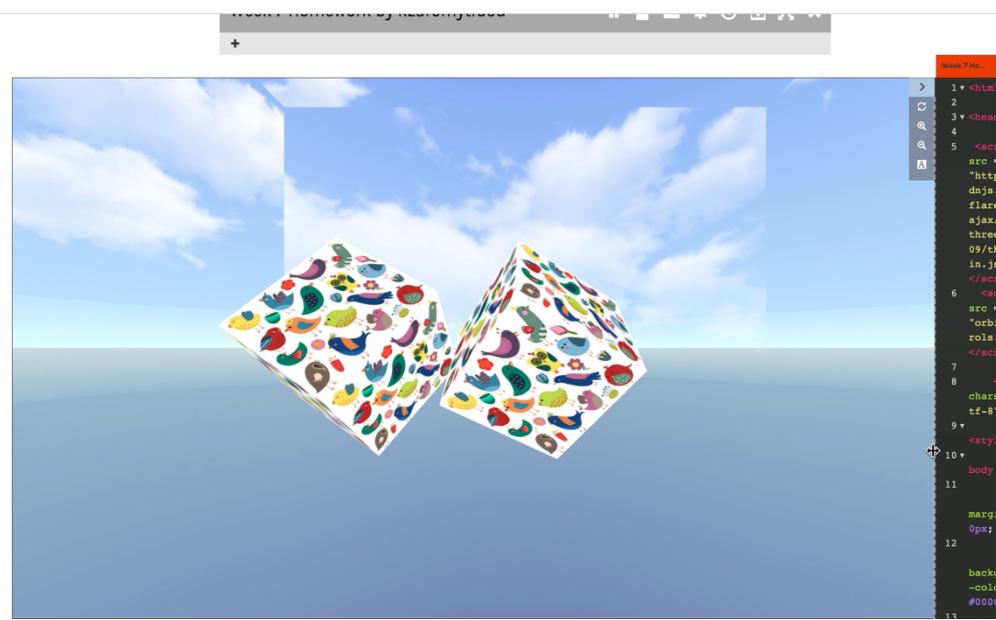
}

</script>

</body>

</html>
```

**Week 7.**



```

<html>

    <head>

        <script src = "https://cdnjs.cloudflare.com/ajax/libs/three.js/109/three.min.js"></script>
        <script src = "orbitControls.js"></script>

        <meta charset="utf-8">
        <style>
            body {
                margin: 0px;
                background-color: #000000;
                overflow: hidden;
            }
        </style>
    </head>

    <body>
        <script>

            // This creates a camera. It has a field of view, a size, a near clipping plane and a far clipping
            plane
            var camera = new THREE.PerspectiveCamera(70, window.innerWidth /
            window.innerHeight, 1, 80);
            // We need to create a scene and add things to it.
            var scene = new THREE.Scene();
            // Now we are going to create some built in geometry
            var geometry = new THREE.BoxGeometry(1, 1, 1);


```

```

var geometry2 = new THREE.BoxGeometry (1,1,1);
// var geometry3 = new THREE.BoxGeometry(1,1,1);

// To do this we need a texture loader object to load the texture
var myTextureLoader = new THREE.TextureLoader();
// Then we can load the texture into a variable
var myTexture = myTextureLoader.load('birds.jpg');

// This defines how the surface of the object reflects light
// We're using Phong. There are lots of other types.
var material = new THREE.MeshBasicMaterial({map: myTexture});

// We can now create a mesh using the geometry and the material
var mesh = new THREE.Mesh(geometry, material);

var mesh2 = new THREE.Mesh(geometry, material);
// var mesh3 = new THREE.Mesh(geometry2, material); // If we want to see stuff, we will
need a light.
// The argument is the colour of the light in hexadecimal.
var light = new THREE.DirectionalLight("rgb(255,255,255)");
// Now we can create our renderer. Thiis renders the scene.
var renderer = new THREE.WebGLRenderer();

// Now we can set some variables for the objects.
camera.position.z = 3;
// Notice we can also us the set method to position things.
//light.position.z = 2;
light.position.set(2,2,2);
// Now we add the mesh and the light to the scene.
scene.add(mesh);
scene.add(mesh2);
scene.add(light);

// This is to make sure that the scene understands the resolution of the device we are on.
renderer.setPixelRatio(window.devicePixelRatio);
// We can also set the size of the render window
renderer.setSize(window.innerWidth, window.innerHeight);
// Finally we want to connect the renderer to the HTML document
document.body.appendChild(renderer.domElement);
// And make sure that when the page is resized, everything gets updated
window.addEventListener('resize', onWindowResize, false);

```

```

var controls = new THREE.OrbitControls(camera, renderer.domElement);

// Now we can have a draw loop.
function draw() {
    mesh.rotation.x += 0.005;
    mesh.rotation.y+= 0.005;
    mesh.position.x = -1.3;

    mesh2.position.x = 0.1;
    mesh2.rotation.x += -0.005;
    mesh2.rotation.y+= -0.005;

//camera.position.x += 0.01;
    controls.update();
    renderer.render(scene, camera,);
    requestAnimationFrame(draw);

//posx.jpg

let materialArray = [];
let texture_ft = new THREE.TextureLoader().load( 'Daylight Box_Front.bmp');
let texture_bk = new THREE.TextureLoader().load( 'Daylight Box_Back.bmp');
let texture_up = new THREE.TextureLoader().load( 'Daylight Box_Top.bmp');
let texture_dn = new THREE.TextureLoader().load( 'Daylight Box_Bottom.bmp');
let texture_rt = new THREE.TextureLoader().load( 'Daylight Box_Right.bmp');
let texture_lf = new THREE.TextureLoader().load('Daylight Box_Left.bmp');

materialArray.push(new THREE.MeshBasicMaterial( { map: texture_ft }));
materialArray.push(new THREE.MeshBasicMaterial( { map: texture_bk }));
materialArray.push(new THREE.MeshBasicMaterial( { map: texture_up }));
materialArray.push(new THREE.MeshBasicMaterial( { map: texture_dn }));
materialArray.push(new THREE.MeshBasicMaterial( { map: texture_rt }));
materialArray.push(new THREE.MeshBasicMaterial( { map: texture_lf }));

for (let i = 0; i < 6; i++)
    materialArray[i].side = THREE.BackSide;
let skyboxGeo = new THREE.BoxGeometry( 10, 10, 10);
let skybox = new THREE.Mesh( skyboxGeo, materialArray );
scene.add( skybox );
animate();

}


```

```

// This is the thing that does the resizing
function onWindowResize() {
    camera.aspect = window.innerWidth /           window.innerHeight;
    camera.updateProjectionMatrix();
    renderer.setSize(window.innerWidth, window.innerHeight);
}

// Finally, call the draw loop.
requestAnimationFrame(draw());


</script>
</body>

</html>

```

## Week 8 1A

The screenshot shows a web browser window titled "Week 8 Homework 1A by k2daronmyt000". The page content displays a blue-to-purple gradient background. To the right of the browser window is a code editor showing the following HTML and JavaScript code:

```

<!DOCTYPE html>
<html>
<head>
<script src =
"https://cdnjs.cloudflare.com/ajax/lib-
e.js/109/three.min.js"></script>
<style>
body {
margin: 0px;
background-color: #000000;
overflow: hidden;
}
</style>
</head>
<body>
<script id="vertexShader" type="x-
shader/x-vertex">
void main() { gl_Position = ve-
position, 1.0; }
</script>
<script id="fragmentShader" type="x-
shader/x-fragment">
//=====
//PUT YOUR GLSL CODE HERE
//=====
precision mediump float;
uniform vec2 resolution;
uniform vec2 mouse;
uniform float time;
void main() {

```

```

<!DOCTYPE
html>

```

```

<html>
<head>
<script src =
"https://cdnjs.cloudflare.com/ajax/libs/three.js/109/three.min.js"></script>
<style>
body {
    margin: 0px;
    background-color: #000000;
    overflow: hidden;
}
</style>
</head>
<body>
<script id="vertexShader" type="x-shader/x-vertex">
void main() { gl_Position = vec4( position, 1.0 ); }
</script>
<script id="fragmentShader" type="x-shader/x-fragment">

//=====
//PUT YOUR GLSL CODE HERE

//=====

precision mediump float;

uniform vec2 resolution;
uniform vec2 mouse;
uniform float time;

void main() {
    vec2 colour = gl_FragCoord.xy/resolution;

    // gl_FragColor = vec4(colour.x,colour.y,0.0,1.0);
    // gl_FragColor = vec4(colour.x,colour.y,4.0,1.0);

    // gl_FragColor = vec4(abs(sin(u_time)),04.3,0.0,1.0);
    // gl_FragColor = vec4(abs(sin(u_time * 4.0)),3.0,0.0,1.0);

    // gl_FragColor = vec4(colour.x*abs(tan(mouse.x)*time),colour.y,3.0,1.0);

    gl_FragColor = vec4(colour.x,colour.y*abs(sin(mouse.x)*time),3.0,4.0);
}

```

```
}
```

```
//=====
//END OF GLSL CODE

//=====
</script>
<script>
    //change the resolution here. 1 is highest
    var pixel_resolution = 5;
    var container, stats;
    var camera, scene, renderer;
    var uniforms;
    init();
    animate();
    function init() {
        camera = new THREE.Camera();
        camera.position.z = 1;
        scene = new THREE.Scene();
        var geometry = new THREE.PlaneBufferGeometry(2, 2);
        uniforms = { time: { type: 'f', value: 1.0 }, resolution: { type:
        'v2', value: new THREE.Vector2() }, mouse: {type: "v2", value: new THREE.Vector2()}};
        var material = new THREE.ShaderMaterial({ uniforms:
        uniforms, vertexShader: document.getElementById('vertexShader').textContent,
        fragmentShader: document.getElementById('fragmentShader').textContent });
        var mesh = new THREE.Mesh(geometry, material);
        scene.add(mesh);
        renderer = new THREE.WebGLRenderer();
        //Hack here to change resolution
        renderer.setPixelRatio(window.devicePixelRatio /
pixel_resolution);
        document.body.appendChild(renderer.domElement);
        onWindowResize();
        window.addEventListener('resize', onWindowResize, false);
        window.addEventListener('mousemove', onMouseMove, false);

    }
    function onWindowResize(event) {
        renderer.setSize(window.innerWidth,
window.innerHeight);
        uniforms.resolution.value.x = renderer.domElement.width;
        uniforms.resolution.value.y = renderer.domElement.height;

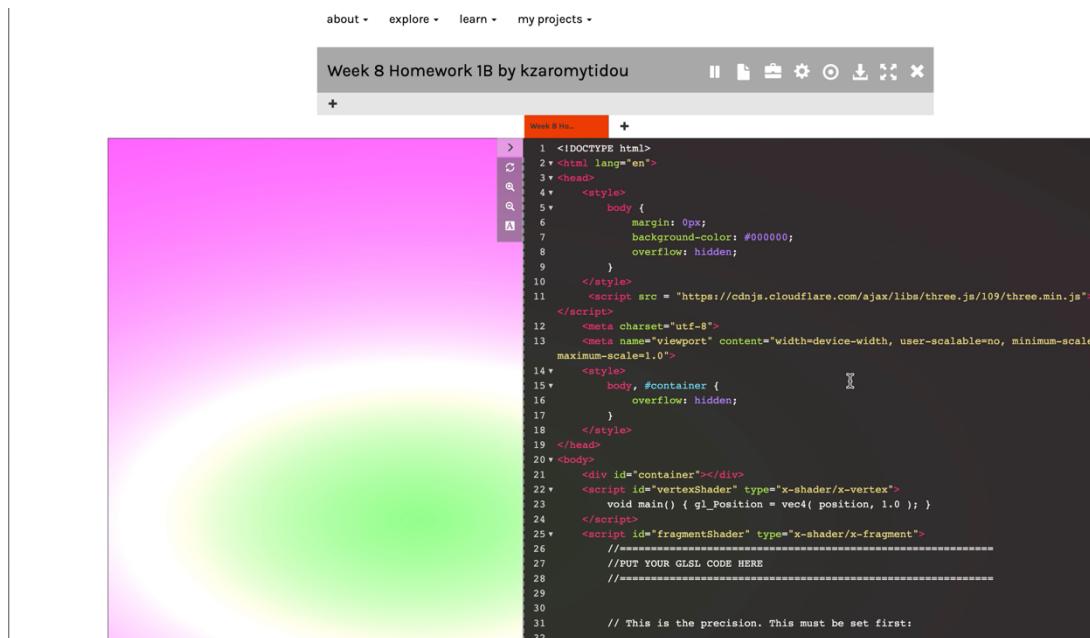
    }
}
```

```

        function animate() {
            requestAnimationFrame(animate);
            render();
        }
        function onMouseMove( event ) {
            uniforms.mouse.value.x = 2 * ( event.clientX / window.innerWidth );
            uniforms.mouse.value.y = 2 * ( 1-(event.clientY) / window.innerHeight
            );
        }
        function render() {
            uniforms.time.value += 0.01;
            renderer.render(scene, camera);
        }
    </script>
</body>
</html>

```

## Week 8 1B



The screenshot shows a code editor with the following code:

```

<!DOCTYPE html>
<html lang="en">
<head>
<style>
body {
    margin: 0px;
    background-color: #000000;
    overflow: hidden;
}
</style>
<script src = "https://cdnjs.cloudflare.com/ajax/libs/three.js/109/three.min.js"></script>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, user-scalable=no, minimum-scale=1.0">
<style>
body, #container {
    overflow: hidden;
}
</style>
</head>
<body>
<div id="container"></div>
<script id="vertexShader" type="x-shader/x-vertex">
void main() { gl_Position = vec4( position, 1.0 ); }
</script>
<script id="fragmentShader" type="x-shader/x-fragment">
//=====
//PUT YOUR GLSL CODE HERE
//=====
// This is the precision. This must be set first:
</script>

```

```

<!DOCTYPE
html>
<html lang="en">
<head>

```

```

<style>
    body {
        margin: 0px;
        background-color: #000000;
        overflow: hidden;
    }
</style>
<script src =
"https://cdnjs.cloudflare.com/ajax/libs/three.js/109/three.min.js"></script>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, user-scalable=no,
minimum-scale=1.0, maximum-scale=1.0">
<style>
    body, #container {
        overflow: hidden;
    }
</style>
</head>
<body>
    <div id="container"></div>
    <script id="vertexShader" type="x-shader/x-vertex">
        void main() { gl_Position = vec4( position, 1.0 ); }
    </script>
    <script id="fragmentShader" type="x-shader/x-fragment">

        //=====
        //PUT YOUR GLSL CODE HERE

        //=====

        // This is the precision. This must be set first:

        precision mediump float;

        //These uniforms need to be set up in your management code:
        uniform vec2 resolution;
        uniform vec2 mouse;
        uniform float time;

        void main() {
            vec2 coord = gl_FragCoord.xy/resolution;
            vec3 color = vec3(1.0,1.0,1.0);
            vec2 translate = vec2(-0.3);

```

```

coord += translate;

color.r = abs( length(coord) - abs(cos(time * 0.07)));
color.g = abs(1.0 + length(coord) - abs(cos(time * 0.01)));
color.b = abs( length(coord) - abs(sin(time * 0.04)));

gl_FragColor = vec4(0.3 / color, 1.0);

}

//=====
//END OF GLSL CODE

=====

</script>
<script>

//change the resolution here. 1 is highest
var pixel_resolution = 3;
var container, stats;
var camera, scene, renderer;
var uniforms;
init();
animate();
function init() {
    container = document.getElementById('container');
    camera = new THREE.Camera();
    camera.position.z = 1;
    scene = new THREE.Scene();
    var geometry = new THREE.PlaneBufferGeometry(2, 2);
    uniforms = { time: { type: 'f', value: 2.0 }, resolution: { type:
'v2', value: new THREE.Vector2() }, mouse: {type: "v2", value: new THREE.Vector2()} };
    var material = new THREE.ShaderMaterial({ uniforms:
uniforms, vertexShader: document.getElementById('vertexShader').textContent,
fragmentShader: document.getElementById('fragmentShader').textContent });
    var mesh = new THREE.Mesh(geometry, material);
    scene.add(mesh);
    renderer = new THREE.WebGLRenderer();
    //Hack here to change resolution
    renderer.setPixelRatio(window.devicePixelRatio /
pixel_resolution);
    container.appendChild(renderer.domElement);
    onWindowResize();
    window.addEventListener('resize', onWindowResize, false);
    window.addEventListener('mousemove', onMouseMove, false);
}

```

```
        }

        function onWindowResize(event) {
            renderer.setSize(window.innerWidth,
                window.innerHeight);
            uniforms.resolution.value.x = renderer.domElement.width;
            uniforms.resolution.value.y = renderer.domElement.height;
        }

    }

    function animate() {
        requestAnimationFrame(animate);
        render();
    }

    function onMouseMove( event ) {
        uniforms.mouse.value.x = 2 * ( event.clientX / window.innerWidth );
        uniforms.mouse.value.y = 2 * ( 1-(event.clientY) / window.innerHeight );
    }

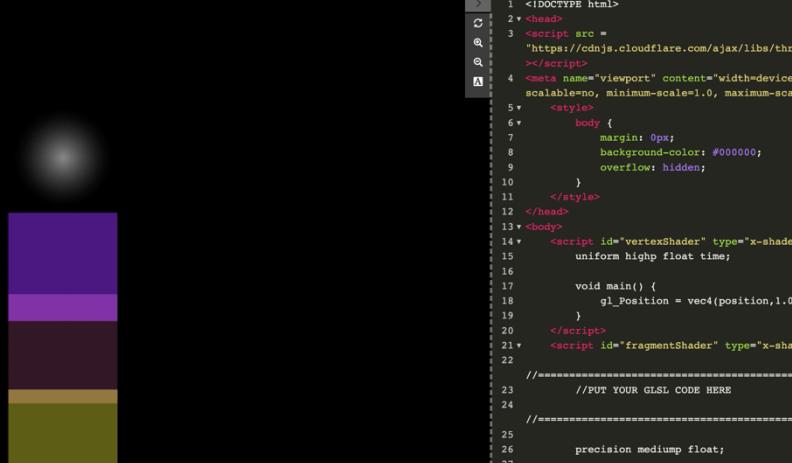
}

function render() {
    uniforms.time.value += 0.01;
    renderer.render(scene, camera);
}

</script>

</body>
</html>
```

**Week 8 2A&B**



The screenshot shows a code editor interface with a tab labeled "GLSL" and a file icon. The code itself is a template for a three.js scene setup. It includes a head section with a script tag pointing to a CDN and a meta tag for the viewport. The body section contains two script tags: one for the vertex shader and one for the fragment shader. Both shaders define a uniform float variable "time". The vertex shader's main function sets the position to a vec4 value. The fragment shader's main function is a placeholder comment //PUT YOUR GLSL CODE HERE. The code ends with several precision statements and uniform declarations for resolution, mouse position, and time.

```
1 <!DOCTYPE html>
2 <head>
3 <script src =
4 "https://cdnjs.cloudflare.com/ajax/libs/three.js/r109/three.js"></script>
5 <meta name="viewport" content="width=device-width, user-
scalable=no, minimum-scale=1.0, maximum-scale=1.0">
6 <style>
7     body {
8         margin: 0px;
9         background-color: #000000;
10        overflow: hidden;
11    }
12 </style>
13 </head>
14 <body>
15 <script id="vertexShader" type="x-shader/x-vertex">
16     uniform highp float time;
17
18     void main() {
19         gl_Position = vec4(position,1.0) ;
20     }
21 </script>
22 <script id="fragmentShader" type="x-shader/x-fragment">
23 //=====
24 //PUT YOUR GLSL CODE HERE
25 //=====
26
27
28
29
30
31
```

```

<!DOCTYPE
html>

<head>
<script src = "https://cdnjs.cloudflare.com/ajax/libs/three.js/109/three.min.js"></script>
<meta name="viewport" content="width=device-width, user-scalable=no, minimum-
scale=1.0, maximum-scale=1.0">
<style>
body {
    margin: 0px;
    background-color: #000000;
    overflow: hidden;
}
</style>
</head>
<body>
<script id="vertexShader" type="x-shader/x-vertex">
uniform highp float time;

void main() {
gl_Position = vec4(position,1.0) ;
}

</script>
<script id="fragmentShader" type="x-shader/x-fragment">

//=====
//PUT YOUR GLSL CODE HERE

//=====

precision mediump float;

uniform vec2 resolution;
uniform vec2 mouse;
uniform highp float time;

float square(vec2 pos, float size) {

    vec2 normCoords = gl_FragCoord.xy/resolution;

    float aspect = resolution.x/resolution.y;
    size*=0.1;
}

```

```
    if (length((normCoords.x-pos.x) * aspect)< size && length(normCoords.y-pos.y) <  
size) {  
  
        return 1.0;  
  
    } else {  
  
        return 0.;  
    }  
}
```

//\_\_\_\_\_ADDING NEW RECT\_\_\_\_\_

```
float square3 (vec2 pos, float size) {  
  
    vec2 normCoords = gl_FragCoord.xy/resolution;  
  
    float aspect = resolution.x/resolution.y;  
    size*=0.2;  
    if (length((normCoords.x-pos.x) * aspect)< size && length(normCoords.y-pos.y) <  
size) {  
  
        return 1.0;  
  
    } else {  
  
        return 0.;  
    }  
}
```

//\_\_\_\_\_

```
float line(vec2 pos, float funct) {  
  
    return step(funct,pos.y)-step(funct,pos.y-0.01);  
}  
  
float circle(vec2 pos, float size) {
```

```

size = 1./size;
size*=10.;

float aspect = resolution.x/resolution.y;

vec2 normCoord = vec2(gl_FragCoord.x/(resolution.x) *
aspect,gl_FragCoord.y/resolution.y);

float colour = distance(normCoord,pos);
return smoothstep(colour * size, colour * size+1.9,1.);
}

void main(){

//vec2 pos = gl_FragCoord.xy/resolution;

float rect = square(vec2(0.15,0.5),1.);

float rect2 = square(vec2(0.15,0.35),1.);

float rect3 = square(vec2(0.15,0.175),1.);
float circleOne = circle(vec2(0.285,0.7),1.);

vec3 squareOne = vec3(0.3,0.1,0.5) * rect;

vec3 squareTwo = vec3(0.2,0.1,0.15) * rect2;

vec3 squareThree = vec3(0.6,0.6,0.15)*abs(sin(time)) * rect3;

//    float x = abs(sin(time)) * 2.;

gl_FragColor = vec4(squareOne + squareTwo + squareThree + circleOne,1.0);
}

=====

//=====
//END OF GLSL CODE

=====

</script>
<script>

//change the resolution here. 1 is highest
var pixel_resolution = 2;

```

```

var stats;
var camera, scene, renderer;
var uniforms;
init();
animate();
function init() {

    camera = new THREE.Camera();
    camera.position.z = 1;
    scene = new THREE.Scene();
    var geometry = new THREE.PlaneBufferGeometry(2,2);
    uniforms = { time: { type: 'f', value: 1.0 }, resolution: { type:
'v2', value: new THREE.Vector2() }, mouse: {type: "v2", value: new THREE.Vector2()}};
    var material = new THREE.ShaderMaterial({ uniforms:
uniforms, vertexShader: document.getElementById('vertexShader').textContent,
fragmentShader: document.getElementById('fragmentShader').textContent });
    var mesh = new THREE.Mesh(geometry, material);
    scene.add(mesh);
    renderer = new THREE.WebGLRenderer();
    //Hack here to change resolution
    renderer.setPixelRatio(window.devicePixelRatio /
pixel_resolution);
    document.body.appendChild(renderer.domElement);
    onWindowResize();
    window.addEventListener('resize', onWindowResize, false);
    window.addEventListener('mousemove', onMouseMove, false);

}

function onWindowResize(event) {
    renderer.setSize(window.innerWidth,
window.innerHeight);
    uniforms.resolution.value.x = renderer.domElement.width;
    uniforms.resolution.value.y = renderer.domElement.height;

}

function animate() {
    requestAnimationFrame(animate);
    render();
}

function onMouseMove( event ) {
    uniforms.mouse.value.x = ( event.clientX / window.innerWidth );
    uniforms.mouse.value.y = ( 1-(event.clientY) / window.innerHeight
);
}

function render() {
    uniforms.time.value += 0.01;
}

```

```

        renderer.render(scene, camera);
    }

```

</script>

</body>

</html>

## Week 8 2C

```

about ▾ explore ▾ learn ▾ my projects ▾

Week 8 HW2C by kzaromytidou
+ [Week 8 HW2C]

115     link.download = filename
116     link.click()
117
118   }
119   function saveString(text, filename) {
120     save(new Blob([text], { type: 'text/plain' }), filename)
121   }
122
123   function exportGLTF() {
124     const exporter = new GLTFExporter()
125     const params = {
126       trs: false,
127       onlyVisible: true,
128       truncatedDrawRange: true,
129       binary: false,
130       maxTextureSize: 4096,
131     }
132
133     exporter.parse(
134       scene,
135       // called when the gltf has been generated
136       function (gltf) {
137         const output = JSON.stringify(gltf, null, 2)
138         console.log(output)
139         saveString(output, 'scene.gltf')
140       },
141       // called when there is an error in the generation
142       function (error) {
143         console.log('An error happened')
144       },
145       params
146     )
147   }
148   exportGLTF()
149 </script>
150 </body>

```

```

<!DOCTYPE
html>

<html lang="en">
<head>
<style>
body {
    margin: 0px;
    background-color: #000000;
    overflow: hidden;
}
</style>
<script src =
"https://cdnjs.cloudflare.com/ajax/libs/three.js/109/three.min.js"></script>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, user-scalable=no,
minimum-scale=1.0, maximum-scale=1.0">
<style>

```

```

        body, #container {
            overflow: hidden;
        }
    </style>
</head>
<body>
    <div id="container"></div>
    <script id="vertexShader" type="x-shader/x-vertex">
        uniform highp float time;

        mat4 scale =
mat4(.9,0,0,0,0,0,0.1,0.01,0,0,0,0,1,0,0.1,1,0,0,0,0.3,0.6,0,0,.3);

        void main() {

            float displacementHeight = .8;
            float displacementY = sin(time + (position.x) * 3.) *
dot(displacementHeight,sin(time)*2.0*tan(time)*0.4);

            vec4 modifiedPosition = vec4(position,2.0);
            modifiedPosition.y += displacementY;
            gl_Position = modifiedPosition * scale ;
        }
    </script>
    <script id="fragmentShader" type="x-shader/x-fragment">

//=====
//PUT YOUR GLSL CODE HERE

//=====

precision mediump float;

uniform vec2 resolution;
uniform vec2 mouse;
uniform highp float time;

void main(){

    vec2 norm_res = vec2(gl_FragCoord.xy / resolution);

    gl_FragColor = vec4(norm_res.x,norm_res.y,3.5,4.5);
}

//=====

```

```

//END OF GLSL CODE

//=====
</script>
<script>

    //change the resolution here. 1 is highest
    var pixel_resolution = 2;
    var container, stats;
    var camera, scene, renderer;
    var uniforms;
    init();
    animate();
    function init() {
        camera = new THREE.Camera();
        camera.position.z = 1.0;
        scene = new THREE.Scene();
        var geometry = new
        THREE.PlaneBufferGeometry(5,2,10,10);//size x, size y, dim x, dim y
        uniforms = { time: { type: 'f', value: 1.0 }, resolution: { type:
        'v2', value: new THREE.Vector2() }, mouse: {type: "v2", value: new THREE.Vector2()}};
        var material = new THREE.ShaderMaterial({ uniforms:
        uniforms, vertexShader: document.getElementById('vertexShader').textContent,
        fragmentShader: document.getElementById('fragmentShader').textContent });
        var mesh = new THREE.Mesh(geometry, material);
        scene.add(mesh);
        renderer = new THREE.WebGLRenderer();
        //Hack here to change resolution
        renderer.setPixelRatio(window.devicePixelRatio /
        pixel_resolution);
        document.body.appendChild(renderer.domElement);
        onWindowResize();
        window.addEventListener('resize', onWindowResize, false);
        window.addEventListener('mousemove', onMouseMove, false);

    }
    function onWindowResize(event) {
        renderer.setSize(window.innerWidth,
        window.innerHeight);
        uniforms.resolution.value.x = renderer.domElement.width;
        uniforms.resolution.value.y = renderer.domElement.height;

    }
    function animate() {
        requestAnimationFrame(animate);
        render();
    }
}

```

```

        function onMouseMove( event ) {
            uniforms.mouse.value.x = 2 * ( event.clientX / window.innerWidth );
            uniforms.mouse.value.y = 2 * ( 1-(event.clientY) / window.innerHeight );
        }
    }

    function render() {
        uniforms.time.value += 0.01;
        renderer.render(scene, camera);
    }
</script>

<script language="javascript" type="text/javascript">
    function save(blob, filename) {
        const link = document.createElement('a')
        link.style.display = 'block'
        document.body.appendChild(link)

        console.log(blob)

        link.href = URL.createObjectURL(blob)
        link.download = filename
        link.click()
    }

    function saveString(text, filename) {
        save(new Blob([text], { type: 'text/plain' }), filename)
    }

    function exportGLTF() {
        const exporter = new GLTFSigner()
        const params = {
            trs: false,
            onlyVisible: true,
            truncateDrawRange: true,
            binary: false,
            maxTextureSize: 4096,
        }

        exporter.parse(
            scene,
            // called when the gltf has been generated
            function (gltf) {
                const output = JSON.stringify(gltf, null, 2)
                console.log(output)
                saveString(output, 'scene.gltf')
            },
        )
    }
</script>

```

```
// called when there is an error in the generation
function (error) {
    console.log('An error happened')
},
params
)
}
exportGLTF()
</script>
</body>
</html>
```