Assignment 4

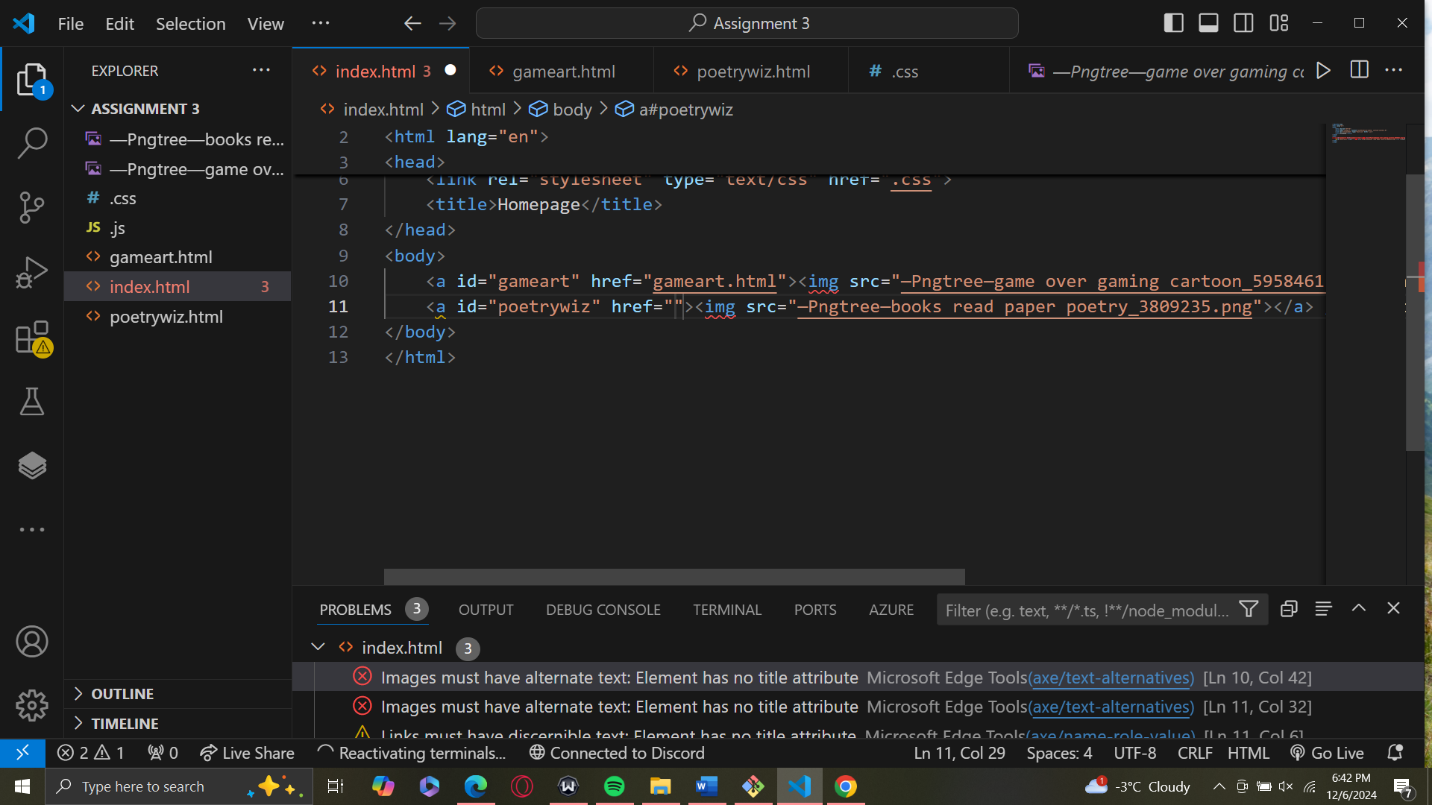
Name: Nihon Anan Rahman

Student ID: - 3210915

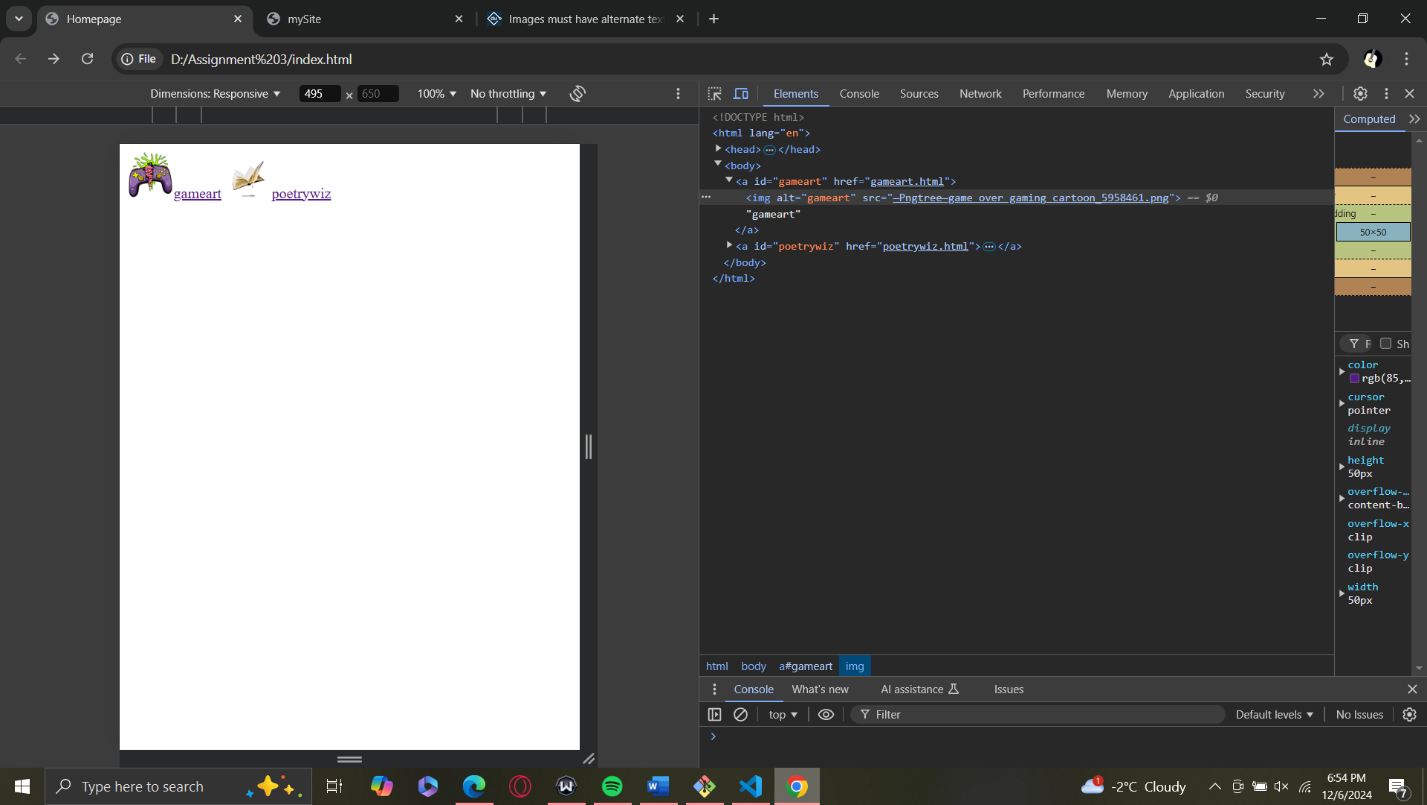
Idea for website: -

A game/ art animation site, with one page being the homepage directing to the other two page where one has pieces of my favorite literature and the other has either an animation art played on loop or an attempt at creating game art.

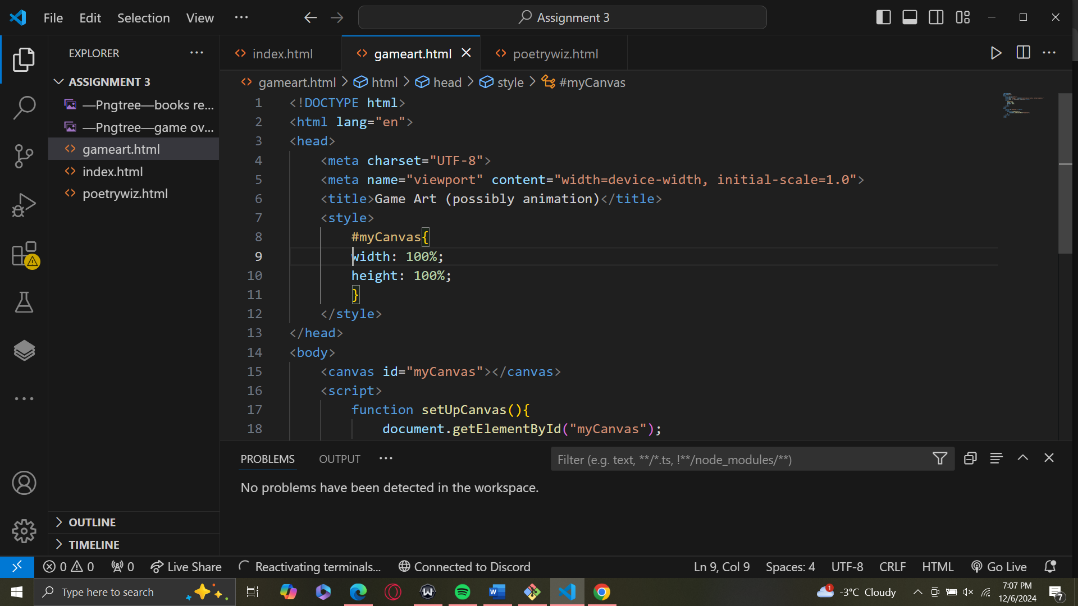
I’ll be using png sources from the following site:- [Game Animation PNG Transparent Images Free Download | Vector Files | Pngtree](https://pngtree.com/so/game-animation)

VS code Programming: -

1). Developing the homepage while using two images as hyperlinks to the other two sites.



## Working on the Game Art animation

putting the <input> tag and <button> tag in the gameart page

## Script Code

[HTML Canvas Lines](https://www.w3schools.com/graphics/canvas_lines.asp)

[Reference](https://p5js.org/reference/)

[Canvas API - Web APIs | MDN](https://developer.mozilla.org/en-US/docs/Web/API/Canvas_API)

[Window: requestAnimationFrame() method - Web APIs | MDN](https://developer.mozilla.org/en-US/docs/Web/API/Window/requestAnimationFrame)

<script>

        var canvas;

        var ctx;

        var w;

        var h;

var b = {

            x: w/2,

            y: h/2,

        }

        var o = {

 x: w/2,

              y: h/2,

        }

        //setting up Canvas function call

        setUpCanvas();

        draw();

 circles(); //will create circles that will uh move from left to right

        function circles(

        }

        function draw(){

             var x = o.x;

            var y = o.y;

            ctx.beginPath();

            ctx.moveTo(o.x,o.y);

            ctx.lineTo(2\*o.x,2\*o.y);

            ctx.closePath();

            ctx.strokeStyle() = 'black';

            ctx.lineWidth = 5;

            ctx.stroke();

        }

        function setUpCanvas(){

            document.getElementById("myCanvas");

            ctx.canvas

            canvas.style.border = "3px solid purple";

            canvas.style.background.color = "sky blue";

            canvas.width = w;

            canvas.height = h;

        }

function random(){

            math.random();

        }

console.log(typeof(b.x));

console.log(`the value of b.x is ${b.x}`);

    </script>

## Script code update: -

I’ll be using the reference below to create shapes that will soon move

[CanvasRenderingContext2D - Web APIs | MDN](https://developer.mozilla.org/en-US/docs/Web/API/CanvasRenderingContext2D) since the one in p5 won’t work the ctx method.

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Game Art (possibly animation)</title>

    <style>

        body{

            background-color: rgb(3, 32, 42);

        }

        #container{

            max-height:fit-content;

            width: fit-content;

            margin: auto;

            text-align: center;

            border: 3px solid purple;

        }

        #myCanvas{

            width: fit-content;

            height: fit-content;

        }

    </style>

</head>

<body>

    <div id="container">

    <canvas id="myCanvas"></canvas>

    </div>

    <label>

        <input type="number" id="nA">

        <button>Change color</button>

    </label>

    <label>

        <input type="number" id="nB">

        <button>Change shape</button>

    </label>

    <script>

        var canvas;

        var ctx;

        var w = 300;

        var h = 150;

        var radius = 5;

        var b = {

            x: w/2,

            y: h/2,

        }

        var x = 300;

        var y = 150;

        var o = {

             x: w/2,

             y: h/2,

        }

        //setting up Canvas function call

        setUpCanvas();

        spongebob(o);

        circles(b); //will create circles that will uh move from left to right

        random(b);

 //create animation of circle

        animationsequence();

        function animationsequence(){

            b.x += 1;

            if(b.x<w){

                b.x = 0;

            }

            //calling the circles function inside here

            circles(b);

        }

        function circles(b){

        var x = b.x;

        var y = b.y;

            ctx.beginPath();

            ctx.arc(x/2, y/2, radius, 0, 2 \* Math.PI);

            ctx.closePath();

            ctx.strokeStyle = 'red';

            ctx.lineWidth = 2;

            ctx.fillStyle = 'purple';

            ctx.fill();

            ctx.stroke();

        }

        function spongebob(o){

            var x = o.x;

            var y = o.y;

            ctx.beginPath();

            ctx.rect(x/50, y/50, (5 + x)/4, (5 + y)/4);   //instead of using square method cause it doesn't exist in the ctx i use rect method

            ctx.closePath();

            ctx.strokeStyle = 'white';

            ctx.lineWidth = 5;

            ctx.fillStyle = "yellow";

            ctx.fill();

            ctx.stroke();

        }

        function setUpCanvas(){

            canvas = document.getElementById("myCanvas"); //forgot to type in 'canvas =....

            ctx = canvas.getContext('2d');

            canvas.style.border = "3px solid purple";

            canvas.style.background = "navy blue";

            canvas.width = w;

            canvas.height = h;

        }

        function random(){

           b.x = Math.floor(Math.random() \* 300);

        }

console.log(typeof(b.x));

console.log(`the value of b.x is ${b.x}`);

    </script>

</body>

</html>

## Updated script code: -

[CanvasRenderingContext2D: clearRect() method - Web APIs | MDN](https://developer.mozilla.org/en-US/docs/Web/API/CanvasRenderingContext2D/clearRect)

[Window: requestAnimationFrame() method - Web APIs | MDN](https://developer.mozilla.org/en-US/docs/Web/API/Window/requestAnimationFrame)

Using the clearrect() method to clear canvas before the animation loops again and the request animationframe method

### Using ChatGPT (from this point on I used GPT with regards to understanding animation used in Canvas): -

<https://chatgpt.com/share/67578daa-8638-800a-8407-70d2421c2cb8>

I’ll also be using this code suggested by Poe to keep track of time of certain object’s coordinates (the circles)

const now = new Date();

console.log('Current Date and Time:', now);

// Get the current timestamp

const timestamp = now.getTime(); // milliseconds since January 1, 1970

console.log('Timestamp:', timestamp);

<script>

        var canvas;

        var ctx;

        var w = 1000;

        var h = 500;

        var radius = 5;

        var b = {

            x: 0,

            y: h/2,

            randomness: rand(2),

            s: 2, //speed of the circle

        }

        var x = 300;

        var y = 150;

        var o = {

             x: w/2,

             y: h/2,

        }

        //setting up Canvas function call

        setUpCanvas();

        spongebob(o);

        circles(b); //will create circles that will uh move from left to right

        rand();

        animate();

        function clear(){

        //to clear the circle before it's moving each time

            ctx.arc(0, 0, 0, 0, 2 \* Math.PI);

            ctx.clearRect(0, 0, canvas.width, canvas.height);

        }

        //

        function animate(){

            clear();

            spongebob(o);

            circles(b);

console.log(`the value of b.x is ${b.x}`);

            requestAnimationFrame(animate);    }

        function circles(b){

            var x = b.x;

            var y = b.y;

            if(b.x<1000){

                b.x+=b.s

            }

         else if(b.x===1000){

            b.x -=b.s;

            }

            ctx.beginPath();

            ctx.arc(x/2, y/2, radius, 0, 2 \* Math.PI);

            ctx.closePath();

            ctx.strokeStyle = 'red';

            ctx.lineWidth = 2;

            ctx.fillStyle = 'purple';

            ctx.fill();

            ctx.stroke();

        }

        function spongebob(o){

            var x = o.x;

            var y = o.y;

            ctx.beginPath();

            ctx.rect(x/25, y/25, (2.5 + x)/8, (2.5 + y)/8);   //instead of using square method cause it doesn't exist in the ctx i use rect method

            ctx.closePath();

            ctx.strokeStyle = 'white';

            ctx.lineWidth = 5;

            ctx.fillStyle = "yellow";

            ctx.fill();

            ctx.stroke();

        }

        function setUpCanvas(){

            canvas = document.getElementById("myCanvas"); //forgot to type in 'canvas =....

            ctx = canvas.getContext('2d');

            canvas.style.border = "3px solid purple";

            canvas.style.background = "navy blue";

            canvas.width = w;

            canvas.height = h;

        }

        function rand(){

           var result = Math.floor(Math.random() \* 1000);

           return result;

           console.log(result);

        }

        let now = new Date(circles(b));

console.log(typeof(b.x));

console.log(`the value of b.x is ${b.x}`);

    </script>

### Further code update: -

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Game Art (possibly animation)</title>

    <style>

        body{

            display: flex;

            align-items: center;

            height: 100vh;

            background-color: rgb(3, 32, 42);

        }

        #container.canvas{

            position: relative; /\* Removed unnecessary properties \*/

            border: 3px solid purple;

        }

    </style>

</head>

<body>

    <div id="container">

    <canvas id="myCanvas" width="1000" height="500"></canvas>

    </div>

    <label>

        <input type="text" id="nA">

        <button>Change color</button>

    </label>

    <label>

        <input type="number" id="nB">

        <button>Change shape</button>

    </label>

    <script>

        var canvas;

        var ctx;

        var w = 1000;

        var h = 500;

        var radius = 5;

        var b = {

            x: 0,

            y: h/2,

            randomness: rand(2),

            s: 2, //speed of the circle

            d: 1 //direction of circle

        }

        // var x = 300;

        // var y = 150;

        const keys=[]; //using array for my keydowns.

        var o = {

             x: w/2,

             y: h/2,

             speedo:5,

             direction:1

        }

        //setting up Canvas function call

        setUpCanvas();

        spongebob(o);

        circles(b); //will create circles that will uh move from left to right

        rand();

         // Create an image object

         var backgroundImage = new Image();

        backgroundImage.src = 'images/background.jpg' // Set the URL of the image

        backgroundImage.onload = function() {

        animate(); // Start animation only after the image is loaded

        };

        //going to addeventlisteners for the keys now

        window.addEventListener("keydown",function(e){

            keys[e.key] = true; //when key held down its set to true to the keys object

        })

        window.addEventListener("keyup",function(e){

            key[e.key] = false; //when key released it's set flase to the keys object

        })

        window.addEventListener("keydown", function(e){

            if(event.key === 'ArrowLeft'){

                o.x -= 2;

            }

        })

        function clear(){

        //to clear the circle before it's moving each time

            ctx.clearRect(0, 0, canvas.width, canvas.height);

        }

        //

        function animate(){

            clear();

            ctx.drawImage(backgroundImage, 0, 0, canvas.width, canvas.height);

            spongebob(o);

            circles(b);

            console.log(`the value of b.x is ${b.x}`); //the value (or distance of b.x is known to me (let's say 1 px = 1 cm))

            requestAnimationFrame(animate);

        }

        function circles(b){

            w = 1000;

            b.x += b.s \* b.d; //x value increments on the basis of speed and direction.

            if(b.x>=w|| b.x<=0){

                b.d \*= -1; //reversing the direction on the basis of this conditon

            }

            ctx.beginPath();

            ctx.arc(b.x, b.y, radius, 0, 2 \* Math.PI);  //the circles finally hit the border now

            ctx.closePath();

            ctx.strokeStyle = 'red';

            ctx.lineWidth = 2;

            ctx.fillStyle = 'purple';

            ctx.fill();

            ctx.stroke();

        }

        function spongebob(o){

            var x = o.x;

            var y = o.y;

            ctx.beginPath();

            ctx.rect(o.x, o.y, (2.5 + o.x)/8, (2.5 + o.y)/8);   //instead of using square method cause it doesn't exist in the ctx i use rect method

            ctx.closePath();

            ctx.strokeStyle = 'white';

            ctx.lineWidth = 5;

            ctx.fillStyle = "yellow";

            ctx.fill();

            ctx.stroke();

            if(keys['ArrowUp']){

                o.y -= o.speedo;  //from the object o 'y' is the y axis

            }

            if(keys['ArrowDown']){

                o.y += o.speedo;

            }

            if(keys['ArrowLeft']){

                o.x -= o.speedo;

            }

            if(keys['ArrowRight']){

                o.x += o.speedo;

            }

        }

        function setUpCanvas(){

            canvas = document.getElementById("myCanvas"); //forgot to type in 'canvas =....

            ctx = canvas.getContext('2d');

            canvas.style.border = "3px solid purple";

            canvas.style.background = "navy blue";

            canvas.width = 1000;

            canvas.height = 500;

        }

        function rand(){

           var result = Math.floor(Math.random() \* 1000);

           return result;

           console.log(result);

        }

        console.log(typeof(b.x));

        console.log(`the value of b.x is ${b.x}`);

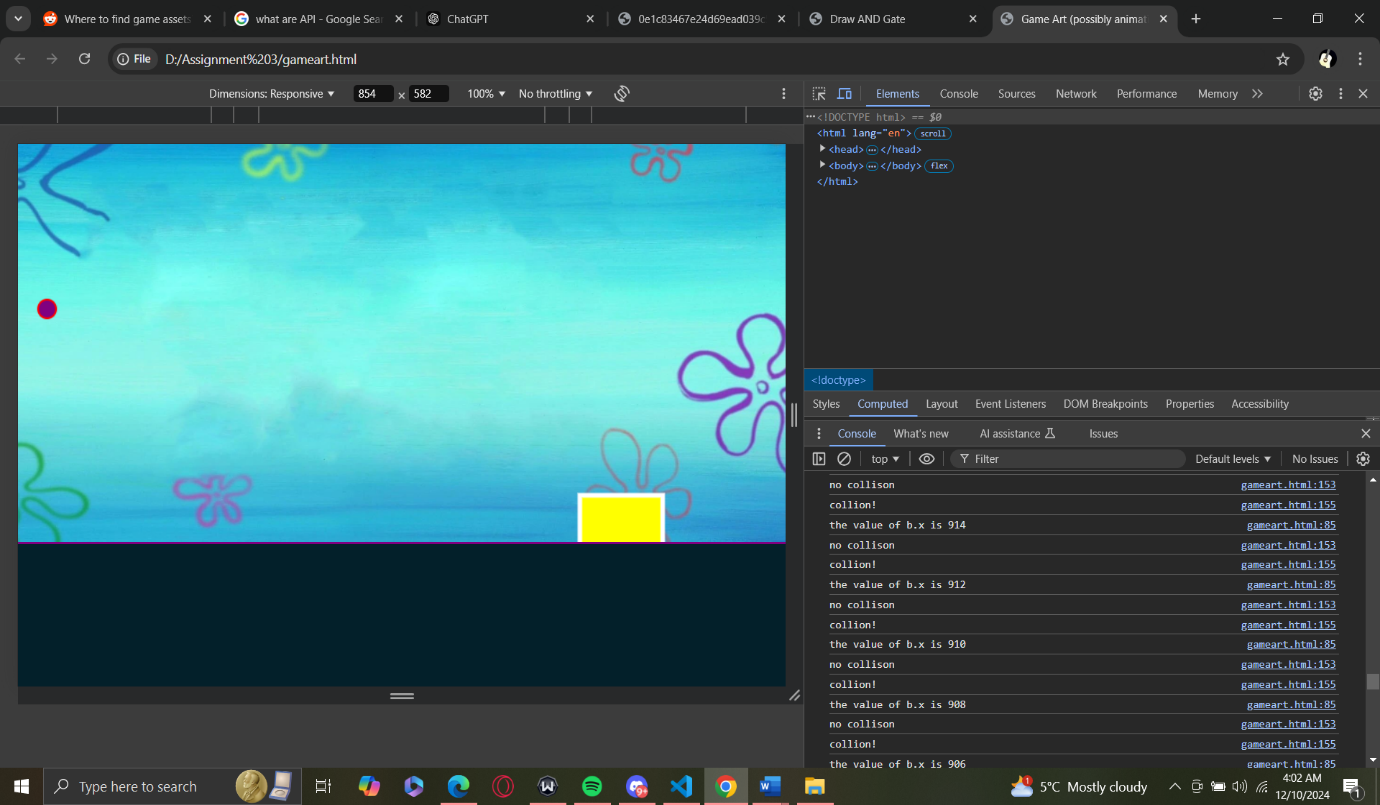
    </script>

    <br>

</body>

</html>

### Collision detection and re-generation of circle after popping



I was able to code this part on my own where I was able to make conditions on the two separate objects (b.x and o.x, b.y and o.y) based on their coordinates

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Game Art (possibly animation)</title>

    <style>

        body{

            display: flex;

            align-items: center;

            height: 100vh;

            background-color: rgb(3, 32, 42);

        }

        #container.canvas{

            position: relative; /\* Removed unnecessary properties \*/

            border: 3px solid purple;

        }

    </style>

</head>

<body>

    <div id="container">

    <canvas id="myCanvas" width="1000" height="500"></canvas>

    </div>

    <label>

        <input type="text" id="nA">

        <button>Change color</button>

    </label>

    <label>

        <input type="number" id="nB">

        <button>Change shape</button>

    </label>

    <script>

        var canvas;

        var ctx;

        var w = 1000;

        var h = 500;

        var radius = 10;

        var b = {

            x: 0,

            y: h/2,

            randomness: rand(2),

            s: 2, //speed of the circle

            d: 1 //direction of circle

        }

        // var x = 300;

        // var y = 150;

        const keys=[]; //using array for my keydowns.

        var o = {

             x: w/2,

             y: h/2,

             speedo:5,

             direction:1

        }

        //setting up Canvas function call

        setUpCanvas();

        spongebob(o);

        circles(b); //will create circles that will uh move from left to right

         // Create an image object

         var backgroundImage = new Image();

        backgroundImage.src = 'images/background.jpg' // Set the URL of the image

        backgroundImage.onload = function() {

        animate(); // Start animation only after the image is loaded

        };

        function clear(){

        //to clear the circle before it's moving each time

            ctx.clearRect(0, 0, canvas.width, canvas.height);

        }

        //

        function animate(){

            clear();

            ctx.drawImage(backgroundImage, 0, 0, canvas.width, canvas.height);

            spongebob(o);

            circles(b);

            DetectCollision();

            console.log(`the value of b.x is ${b.x}`); //the value (or distance of b.x is known to me (let's say 1 px = 1 cm))

            requestAnimationFrame(animate);

        }

        function circles(b){

            w = 1000;

             b.x += b.s \* b.d; //x value increments on the basis of speed and direction.

            b.y += b.s \* b.d;

            if((b.x>=w|| b.x<=0) && (b.y<= h|| b.y>=0)){

                b.d \*= -1; //reversing the direction on the basis of this conditon

                b.d \*=  1;

            }

//update:- I want the circle to move in the y- direction too

            ctx.beginPath();

            ctx.arc(b.x, b.y, radius, 0, 2 \* Math.PI);  //the circles finally hit the border now

            ctx.closePath();

            ctx.strokeStyle = 'red';

            ctx.lineWidth = 2;

            ctx.fillStyle = 'purple';

            ctx.fill();

            ctx.stroke();

        }

    // Add event listeners for key presses

    window.addEventListener("keydown", function (e) {

        keys[e.key] = true;

    });

    window.addEventListener("keyup", function (e) {

        keys[e.key] = false;

    });

        function spongebob(o){

            var x = o.x;

            var y = o.y;

            ctx.beginPath();

            ctx.rect(o.x, o.y, (2.5 + o.x)/8, (2.5 + o.y)/8);   //instead of using square method cause it doesn't exist in the ctx i use rect method

            ctx.closePath();

            ctx.strokeStyle = 'white';

            ctx.lineWidth = 5;

            ctx.fillStyle = "yellow";

            ctx.fill();

            ctx.stroke();

            if (keys['ArrowLeft']) {

            o.x -= o.speedo; // Move left

        }

        if (keys['ArrowRight']) {

            o.x += o.speedo; // Move right

        }

        if (keys['ArrowUp']) {

            o.y -= o.speedo; // Move up

        }

        if (keys['ArrowDown']) {

            o.y += o.speedo; // Move down

        }

        }

        //detect collision

        function DetectCollision(){

            var stouchCirclex = b.x - o.x;

            var stouchCircley = b.y - o.y; //to get the resultant of the x and y coordinates of both circle and spongbob

                                            //if resultant is zero it collided so the following condition will be

            if(stouchCirclex > (radius + 2.5 + o.x)/8){

                console.log('no collison') //since resultant is greater than the combined x coordinates

            } else(stouchCirclex = (radius + 2.5 + o.x)/8)

                console.log('collion!');

            //if proven false

        }

### Updated code (where spongebob finally moves): -

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Game Art (possibly animation)</title>

    <style>

        body{

            display: flex;

            align-items: center;

            height: 100vh;

            background-color: rgb(3, 32, 42);

        }

        #container.canvas{

            position: relative; /\* Removed unnecessary properties \*/

            border: 3px solid purple;

        }

    </style>

</head>

<body>

    <div id="container">

    <canvas id="myCanvas" width="1000" height="500"></canvas>

    </div>

    <label>

        <input type="text" id="nA">

        <button>Change color</button>

    </label>

    <label>

        <input type="number" id="nB">

        <button>Change shape</button>

    </label>

    <script>

        var canvas;

        var ctx;

        var w = 1000;

        var h = 500;

        var radius = 5;

        var b = {

            x: 0,

            y: h/2,

            randomness: rand(2),

            s: 2, //speed of the circle

            d: 1 //direction of circle

        }

        // var x = 300;

        // var y = 150;

        const keys=[]; //using array for my keydowns.

        var o = {

             x: w/2,

             y: h/2,

             speedo:5,

             direction:1

        }

        //setting up Canvas function call

        setUpCanvas();

        spongebob(o);

        circles(b); //will create circles that will uh move from left to right

        rand();

         // Create an image object

         var backgroundImage = new Image();

        backgroundImage.src = 'images/background.jpg' // Set the URL of the image

        backgroundImage.onload = function() {

        animate(); // Start animation only after the image is loaded

        };

        function clear(){

        //to clear the circle before it's moving each time

            ctx.clearRect(0, 0, canvas.width, canvas.height);

        }

        //

        function animate(){

            clear();

            ctx.drawImage(backgroundImage, 0, 0, canvas.width, canvas.height);

            spongebob(o);

            moveSpongebob(o);

            circles(b);

            console.log(`the value of b.x is ${b.x}`); //the value (or distance of b.x is known to me (let's say 1 px = 1 cm))

            requestAnimationFrame(animate);

        }

        function circles(b){

            w = 1000;

            b.x += b.s \* b.d; //x value increments on the basis of speed and direction.

            if(b.x>=w|| b.x<=0){

                b.d \*= -1; //reversing the direction on the basis of this conditon

            }

            ctx.beginPath();

            ctx.arc(b.x, b.y, radius, 0, 2 \* Math.PI);  //the circles finally hit the border now

            ctx.closePath();

            ctx.strokeStyle = 'red';

            ctx.lineWidth = 2;

            ctx.fillStyle = 'purple';

            ctx.fill();

            ctx.stroke();

        }

               //going to addeventlisteners for the keys now

               function moveSpongebob(o) {

        if (keys['ArrowLeft']) {

            o.x -= o.speedo; // Move left

        }

        if (keys['ArrowRight']) {

            o.x += o.speedo; // Move right

        }

        if (keys['ArrowUp']) {

            o.y -= o.speedo; // Move up

        }

        if (keys['ArrowDown']) {

            o.y += o.speedo; // Move down

        }

    }

    // Add event listeners for key presses

    window.addEventListener("keydown", function (e) {

        keys[e.key] = true;

    });

    window.addEventListener("keyup", function (e) {

        keys[e.key] = false;

    });

    console.log(keys)

        function spongebob(o){

            var x = o.x;

            var y = o.y;

            ctx.beginPath();

            ctx.rect(o.x, o.y, (2.5 + o.x)/8, (2.5 + o.y)/8);   //instead of using square method cause it doesn't exist in the ctx i use rect method

            ctx.closePath();

            ctx.strokeStyle = 'white';

            ctx.lineWidth = 5;

            ctx.fillStyle = "yellow";

            ctx.fill();

            ctx.stroke();

            function moveSpongebob(o) {

        if (keys['ArrowLeft']) {

            o.x -= o.speedo; // Move left

        }

        if (keys['ArrowRight']) {

            o.x += o.speedo; // Move right

        }

        if (keys['ArrowUp']) {

            o.y -= o.speedo; // Move up

        }

        if (keys['ArrowDown']) {

            o.y += o.speedo; // Move down

        }

    }

        }

        function setUpCanvas(){

            canvas = document.getElementById("myCanvas"); //forgot to type in 'canvas =....

            ctx = canvas.getContext('2d');

            canvas.style.border = "3px solid purple";

            canvas.style.background = "navy blue";

            canvas.width = 1000;

            canvas.height = 500;

        }

        function rand(){

           var result = Math.floor(Math.random() \* 1000);

           return result;

           console.log(result);

        }

        console.log(typeof(b.x));

        console.log(`the value of b.x is ${b.x}`);

    </script>

    <br>

</body>

</html>

### More updates on script: -

<https://poe.com/s/ulhmWPLZGXNkn0i4mBb1> (I asked Poe about the why my circle kept going outside the canvas boundary)

        function circles(b){

            w = 1000;

            var randx = Math.floor(Math.random()\*canvas.width);

            var randy = Math.floor(Math.random()\*canvas.height);

            b.x += (b.s \* b.d)\*randx;

            b.y += (b.s \* b.d)\*randy; //x value increments on the basis of speed and direction.

            if((b.x>=w|| b.x<=0)){

                b.d \*= -1; //reversing the direction on the basis of this conditon

            }

            if (b.y >= h || b.y <= 0) {

            b.y = (b.s \* b.d)\*randy; // Clamp to bounds

          }

            ctx.beginPath();

            ctx.arc(b.x, b.y, radius, 0, 2 \* Math.PI);  //the circles finally hit the border now

            ctx.closePath();

            ctx.strokeStyle = 'red';

            ctx.lineWidth = 2;

            ctx.fillStyle = 'purple';

            ctx.fill();

            ctx.stroke();

        }

        //detect collision

        function DetectCollision(){

            var stouchCirclex = b.x - o.x;

            var stouchCircley = b.y - o.y; //to get the resultant of the x and y coordinates of both circle and spongbob

                                            //if resultant is zero it collided so the following condition will be

            if(stouchCirclex > (radius + o.x)/8 && stouchCircley > (radius + o.y)/8){

                radius = 5;

                console.log('no collison'); //since resultant is greater than the combined x coordinates

            } else if(stouchCirclex <=  radius + (o.x)/8 && stouchCircley <= radius + (o.y/8)){

                radius = 0;

                console.log('collision!');

             }

            //if proven false

        }

        function setUpCanvas(){

            canvas = document.getElementById("myCanvas"); //forgot to type in 'canvas =....

            ctx = canvas.getContext('2d');

            canvas.style.border = "3px solid purple";

            canvas.style.background = "navy blue";

            canvas.width = 1000;

            canvas.height = 500;

        }

        function rand(){

           var result = Math.floor(Math.random() \* 1000);

           return result;

           console.log(result);

        }

        console.log(typeof(b.x));

        console.log(`the value of b.x is ${b.x}`);

    </script>

    <br>

</body>

</html>

### Issues with the following code: -

//conditional for changing fillStyle

            document.getElementById('nA').addEventListener("click", function() {

            var changeColor = document.getElementById('nA').value.trim(); // Use .value to get the input's value

                 if (b.c.includes(changeColor)) {

                    b.c[0] = changeColor;

                    ctx.fillStyle = changeColor; // Update the first color in the array

                    console.log(`changed color to ${b.c}`);

             } else {

                ctx.fillStyle = 'purple';

             }

            });

//another conditional for adding new colors to the array

            document.getElementById('nB').addEventListener("click", function(){

            var newcolor = document.getElementById("nB").value.trim();

            if (newcolor === b.c){

                b.c.push(newcolor); //when found false that the new color isn't in the array

                console.log(b.c.push(newcolor));

                console.log(b.c);

            }

        })

Despite using Poe for help with this segment, I couldn’t resolve it.

Key Issues: -

I put the eventlistener to not the button.

Resolution to the problem: -

                   //conditional for changing fillStyle

  document.getElementById('inputnA').addEventListener("click", function(circles) {

            var changeColor = document.getElementById('nA').value.trim(); // Use .value to get the input's value

                 if (b.c.includes(changeColor)) {

                    b.c[0] = changeColor;

                    ctx.fillStyle = changeColor; // Update the first color in the array

                    console.log(`changed color to ${b.c}`);

             } else {

                b.c[0] = 'purple';

             }

            });

            //another conditional for adding new colors to the array

            document.getElementById('inputnB').addEventListener("click", function(circles){

            var newcolor = document.getElementById("nB").value.trim();

            if (!b.c.includes(changeColor)){  //can't use === with an array where i newcolor === b.c

                b.c.push(newcolor) = b.c[0]; //when found false that the new color isn't in the array

                console.log(b.c.push(newcolor));

                console.log(b.c);

            }

        })

# GameArt Insiration + Webdesign:-

[(13) How to Make a Game with JavaScript and HTML Canvas | Keyboard Input & Sprite Animation [Vanilla JS] - YouTube](https://www.youtube.com/watch?v=EYf_JwzwTlQ)

<https://poe.com/s/fk8eHBqoEROmnZpkSlHq> (I asked poe how I could add a background image to my canvas).

A website that I referenced for this assignment is one of my previous own: -

[MeghOmala](https://dai-droid.github.io/)

A large portion of coding for the game art that I coded was based on what I had learnt in class and my own research from it from sites like, w3cschools, and code academy from where I had notes about JavaScript canvas.

# Poetrywiz(page)

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Poems</title>

    <style>

        body {

            font-family: Arial, sans-serif;

            margin: 20px;

        }

        label {

            display: block;

            margin-bottom: 10px;

            border-width: 2px;

            border-style: dotted;

            border-color: blue;

            padding: 10px;

        }

        #generatepoetry {

            display: block;

            margin-top: 20px;

            padding: 10px;

            border: 1px solid pink;

            min-height: 50px;

        }

        button {

            margin-left: 10px;

        }

        body{

            background-color: bisque;

        }

        h1{

            font-style: italic;

        }

    </style>

</head>

<body>

    <h1>Poetry</h1>

    <label>

        <input id="insertp" placeholder="Enter your poem here" />

        <button onclick="addPoem()">Enter Poem</button>

    </label>

    <label>

        <output id="generatepoetry"></output>

    </label>

    <script>

        //i did not use canvas for this one

        function addPoem() {

            const poemInput = document.getElementById('insertp').value;

            const outputArea = document.getElementById('generatepoetry');

            outputArea.textContent = poemInput;

            document.getElementById('insertp').value = '';

        }

    </script>

</body>

</html>

# Html code :-

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Homepage</title>

    <style>

      body {

        background-color:rgb(33, 4, 63);

        justify-content: space-evenly;

        align-items: center;

        flex-wrap: wrap;

        height: 100%;

        width: 50%;

      }

a img{

    height: 500px;

    width: 500;

}

#gameart {

    background-color: rgba(255, 255, 255, 0.8); /\* Light background for contrast \*/

    border-radius: 8px; /\* Rounded corners \*/

    padding: 10px; /\* Padding around the text and image \*/

    transition: transform 0.3s, background-color 0.3s; /\* Smooth transition effects \*/

}

#gameart:hover {

    transform: scale(1.05); /\* Slightly enlarge on hover \*/

    background-color: rgb(238, 120, 228); /\* Change background on hover \*/

}

#poetrywiz{

    background-color: rgba(255, 255, 255, 0.8); /\* Light background for contrast \*/

    border-radius: 8px;

    padding: 10px;

    transition: transform 0.3s, background-color 0.3s;

}

#poetrywiz:hover {

    transform: scale(1.05);

    background-color: rgb(203, 194, 212);

}

h1{

    text-align: center;

    color: aqua;

}

#div1{

    align-items: center;

}

#div2{

    align-items: center;

}

nav:hover{

    transform: scale(0.5);

}

    </style>

</head>

<h1>Gameart + Poetry</h1>

<body img="\images\Untitled2-ezgif.com-video-to-webp-converter.webp">

    <div id="nav">

       <div1 id="div1"><a id="gameart" href="gameart.html"><img alt="gameart" src="—Pngtree—game over gaming cartoon\_5958461.png">gameart</a> </div1>

     <div2 id="div2"> <a id="poetrywiz" href="poetrywiz.html"><img alt="poetry" src="—Pngtree—books read paper poetry\_3809235.png">poetrywiz</a> </div2>

    </div>

</body>

</html>

## I asked poe to center properly as I tried to do it: -

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Homepage</title>

<style>

body {

background-color: rgb(33, 4, 63);

display: flex;

justify-content: space-evenly; /\* Even spacing \*/

align-items: center;

flex-wrap: wrap;

height: 100vh; /\* Full viewport height \*/

margin: 0; /\* Remove default margin \*/

}

a img {

height: 300px; /\* Adjust height as needed \*/

width: 300px; /\* Ensure width is defined \*/

}

#nav {

display: flex; /\* Use flexbox layout \*/

justify-content: space-around; /\* Even spacing between items \*/

width: 100%; /\* Full width for nav \*/

}

#gameart, #poetrywiz {

background-color: rgba(255, 255, 255, 0.8); /\* Light background for contrast \*/

border-radius: 8px; /\* Rounded corners \*/

padding: 10px; /\* Padding around the text and image \*/

transition: transform 0.3s, background-color 0.3s; /\* Smooth transition effects \*/

text-align: center; /\* Center text \*/

flex: 1; /\* Allow items to grow \*/

margin: 10px; /\* Space between items \*/

}

#gameart:hover {

transform: scale(1.05); /\* Slightly enlarge on hover \*/

background-color: rgb(238, 120, 228); /\* Change background on hover \*/

}

#poetrywiz:hover {

transform: scale(1.05);

background-color: rgb(203, 194, 212);

}

h1 {

text-align: center;

color: aqua;

margin: 20px 0; /\* Space above and below the heading \*/

}

</style>

</head>

<body>

<h1>Gameart + Poetry</h1>

<div id="nav">

<a id="gameart" href="gameart.html">

<img alt="gameart" src="—Pngtree—game over gaming cartoon\_5958461.png">

gameart

</a>

<a id="poetrywiz" href="poetrywiz.html">

<img alt="poetry" src="—Pngtree—books read paper poetry\_3809235.png">

poetrywiz

</a>

</div>

</body>

</html>

# Summary: -

## Poe and ChatGPT: -

I have mentioned and given the links of when I needed other references of how I could lay out the code in the comments. It was useful as it pointed out parts where I may have made syntax errors, but sometimes if I tried to copy and paste the entire code; the game wouldn’t function the way I initially designed it to be. This may be because I wasn’t always very specific or linear with my thought processes and code.

I think AI only works as a tool, and it’s better to work with our own rationale when thinking about the logic behind a game design whether visually or technically.

I mostly used code from Poe for my second webpage, ‘Poetrywiz’, and wanted to expand on that by creating a database of poems collected based on the inputs of poems by the users. For now, it only generates the input in its output.

So, while Poe Assistant and ChatGPT were useful when I needed to remember a method in JavaScript or if there were any new ones that could help animate my game. Most of basis of it was coded by me, and a lot of the conditionals I improved, and at times I debugged some errors myself; but ChatGPT definitely smoothed that process for me. It saves time, and helps with the CSS styling that you think of.