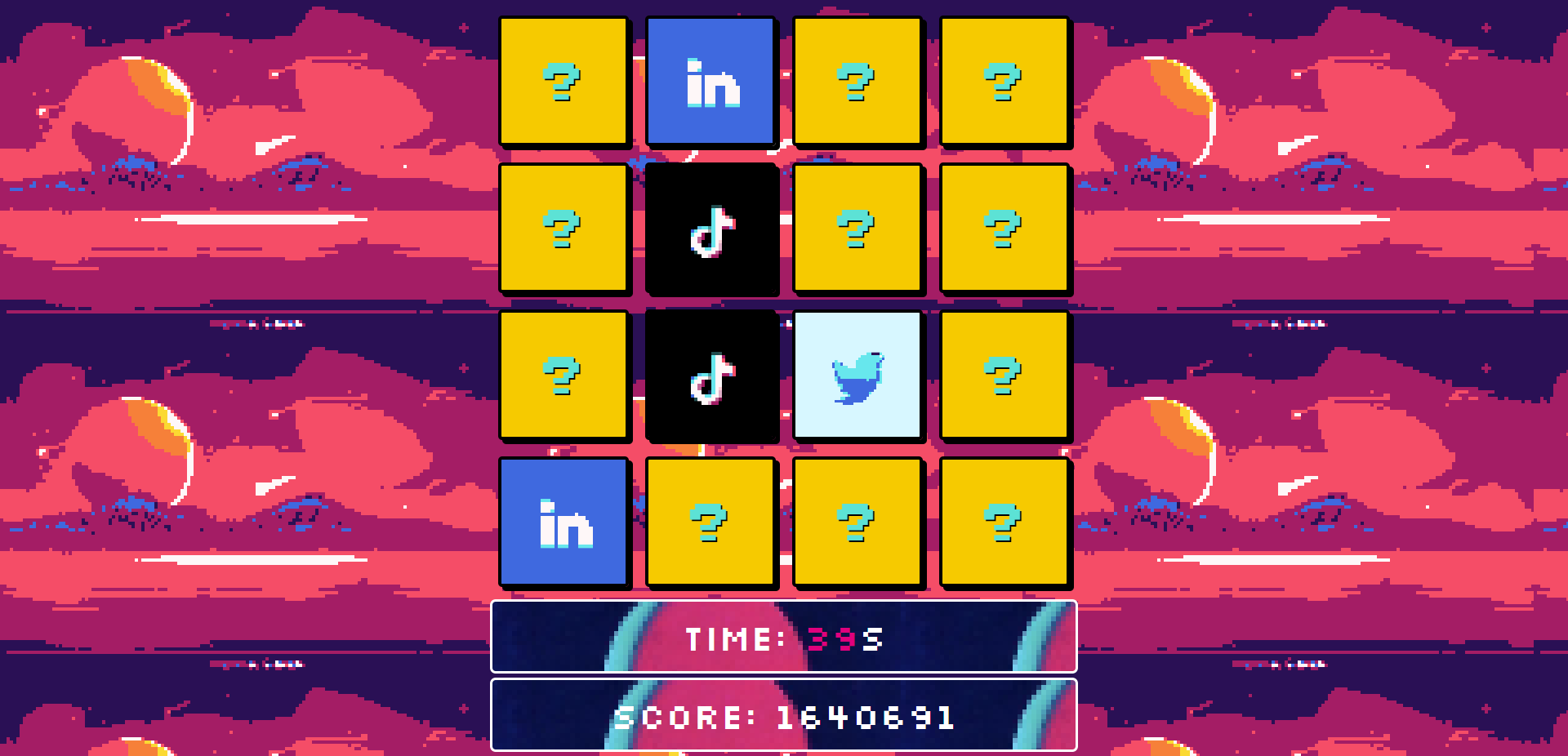
**HOME PAGE**

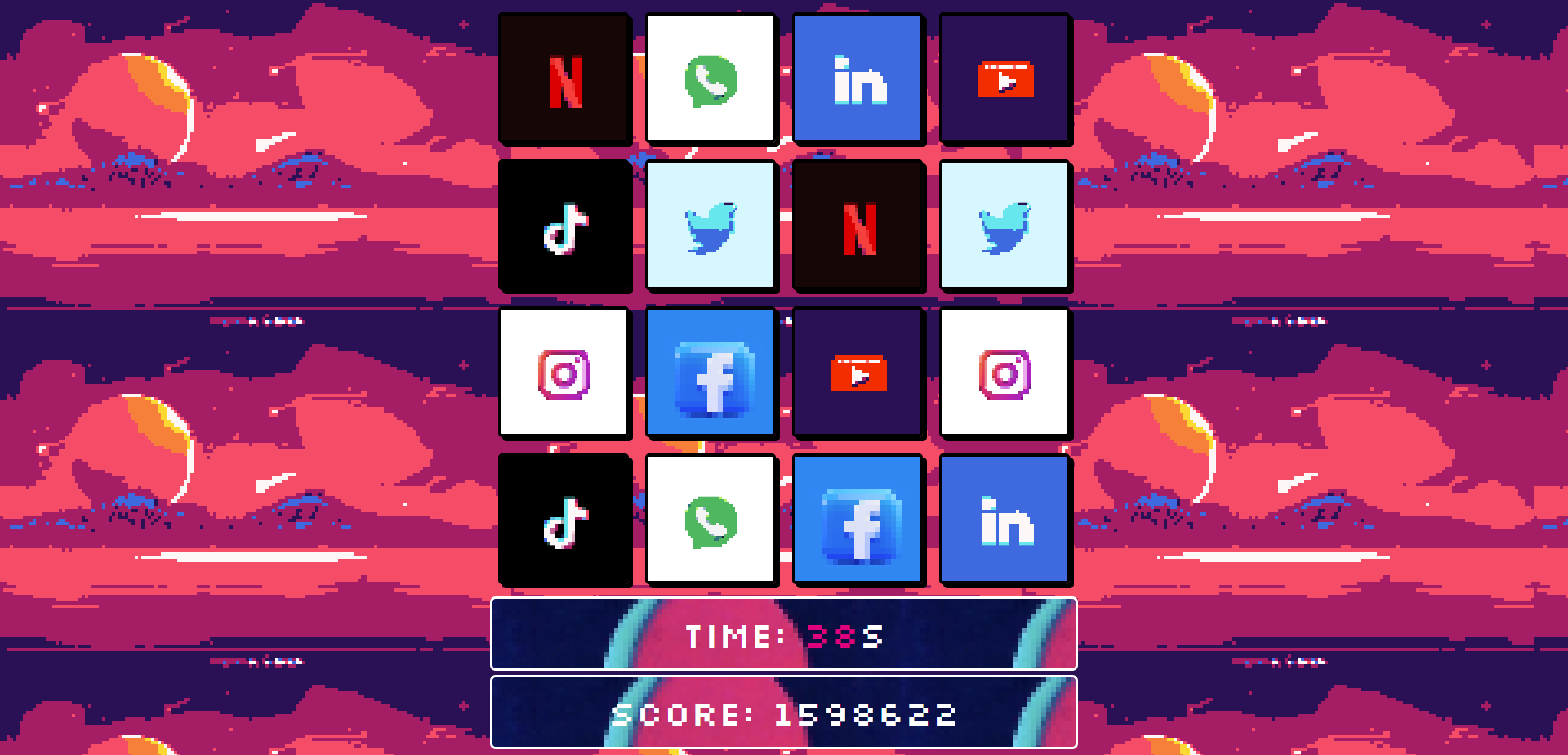
****

****

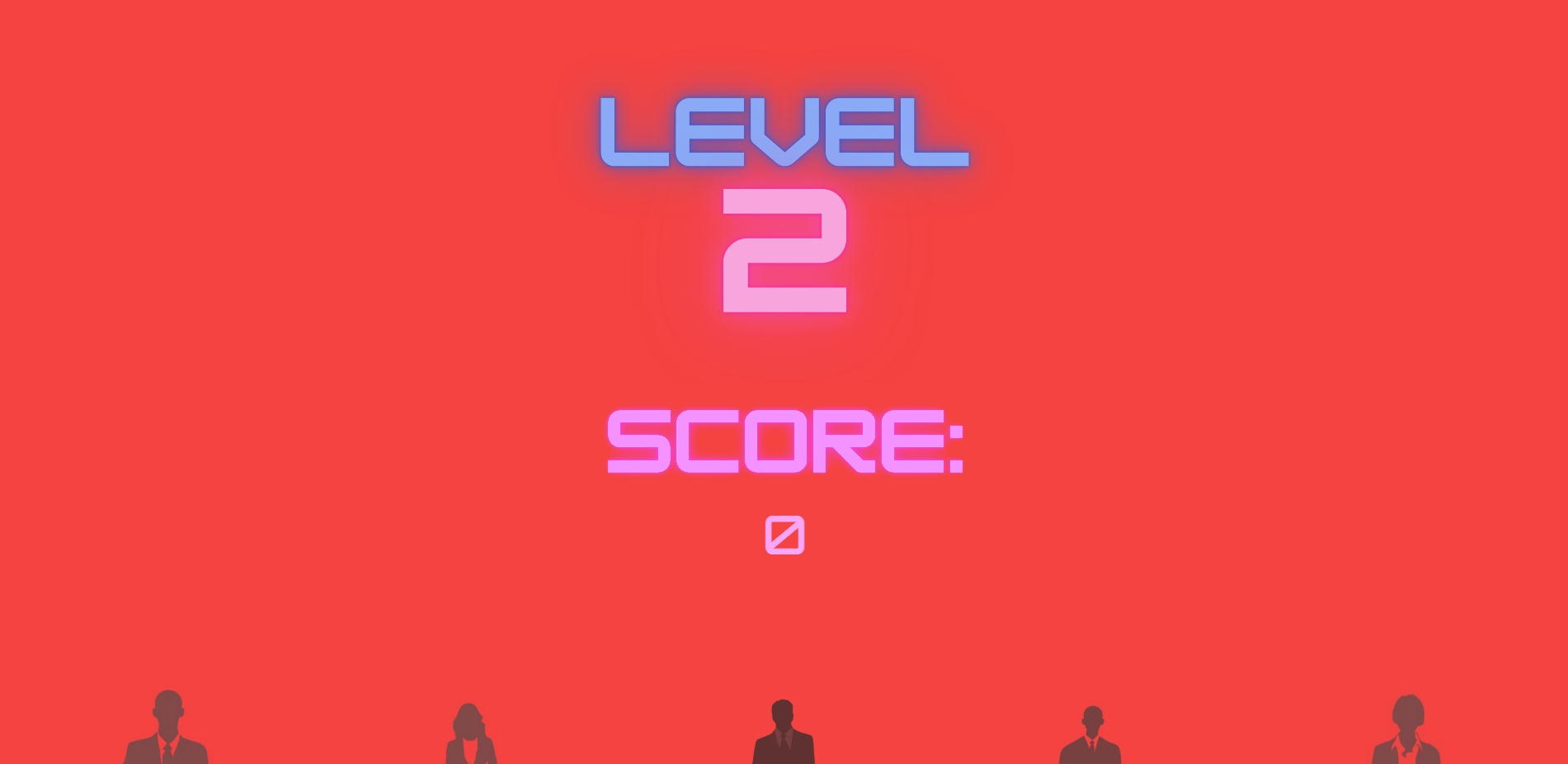
****

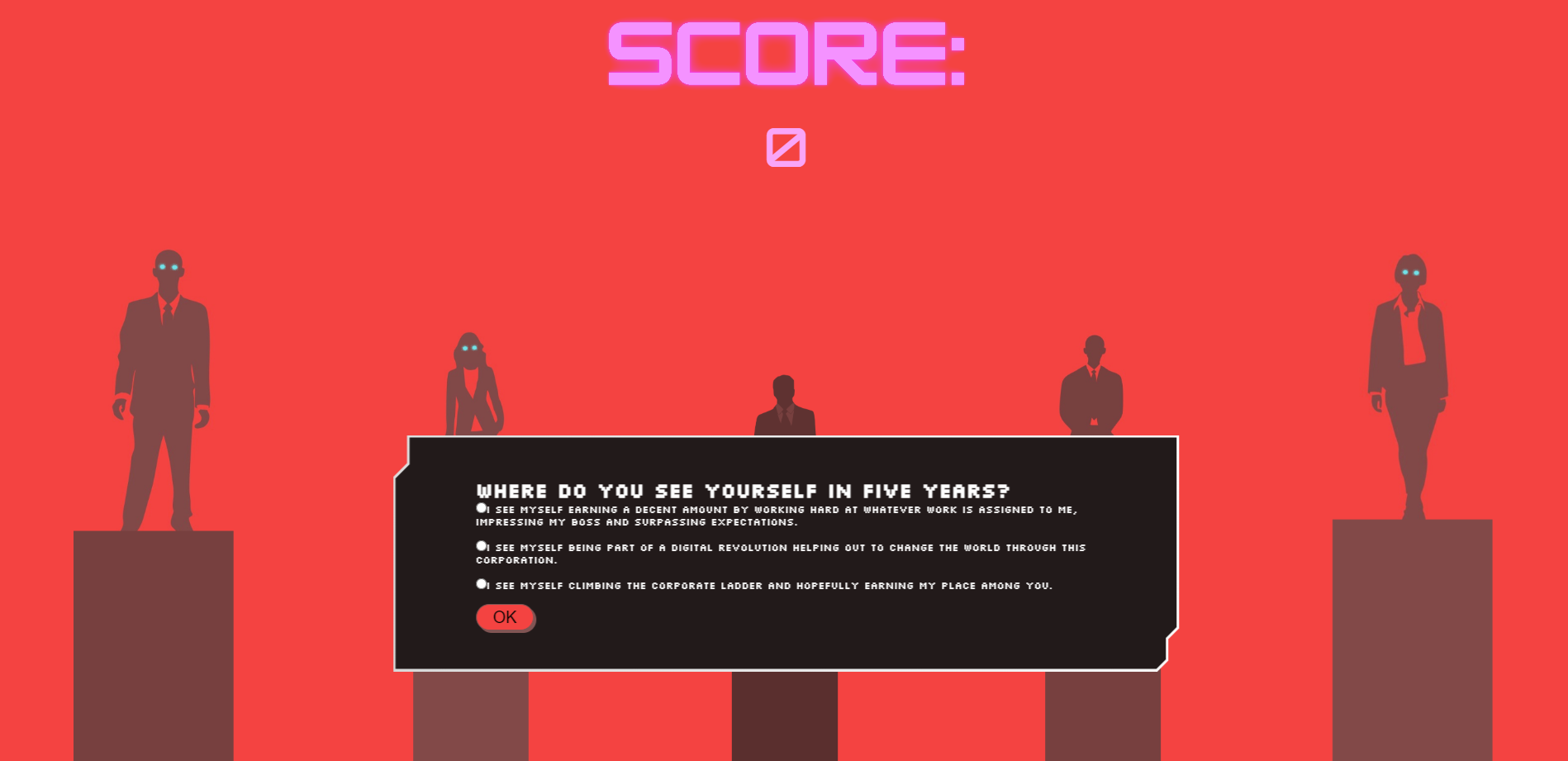
**LEVEL 1: Card Matching Game**

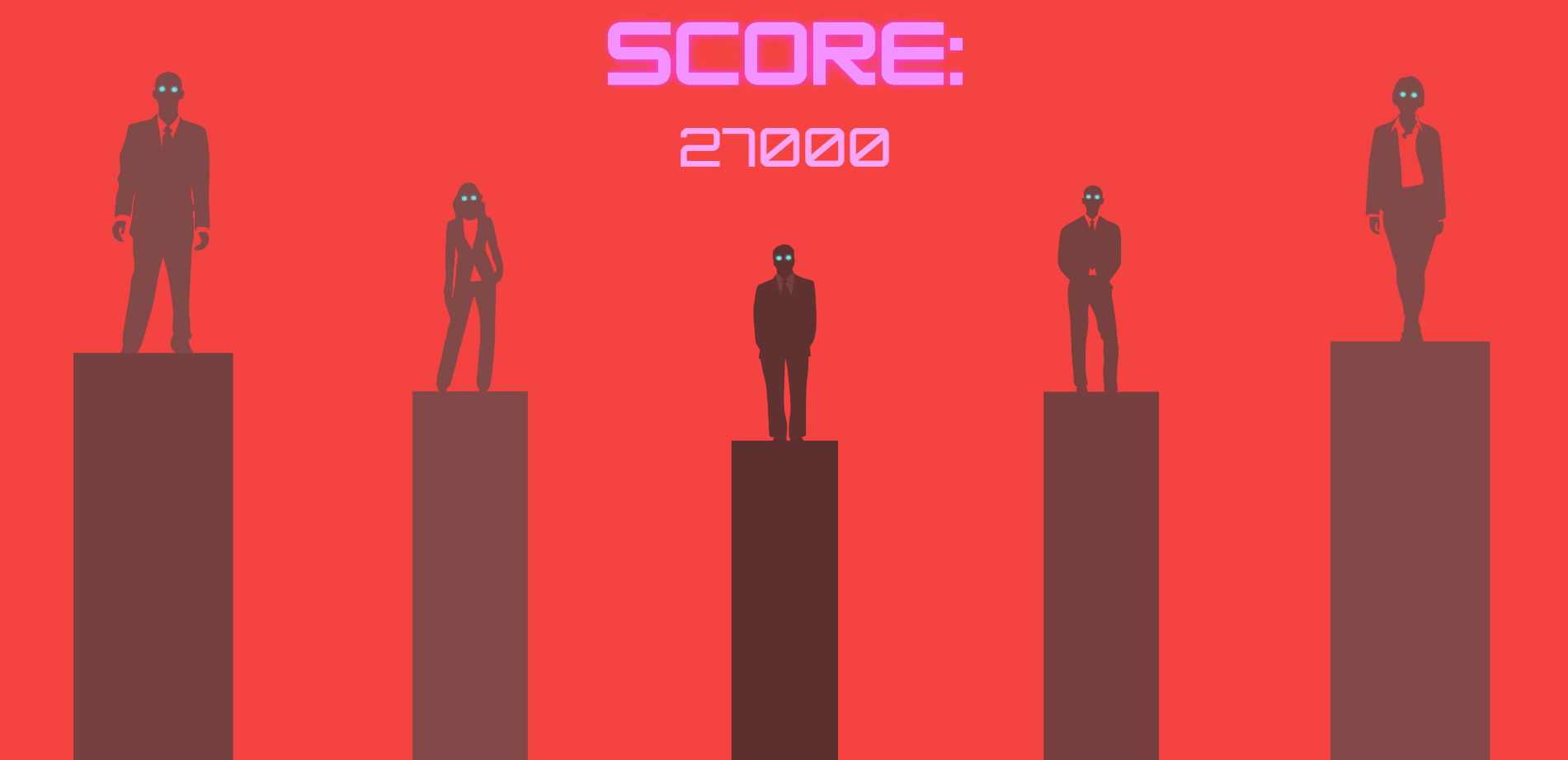




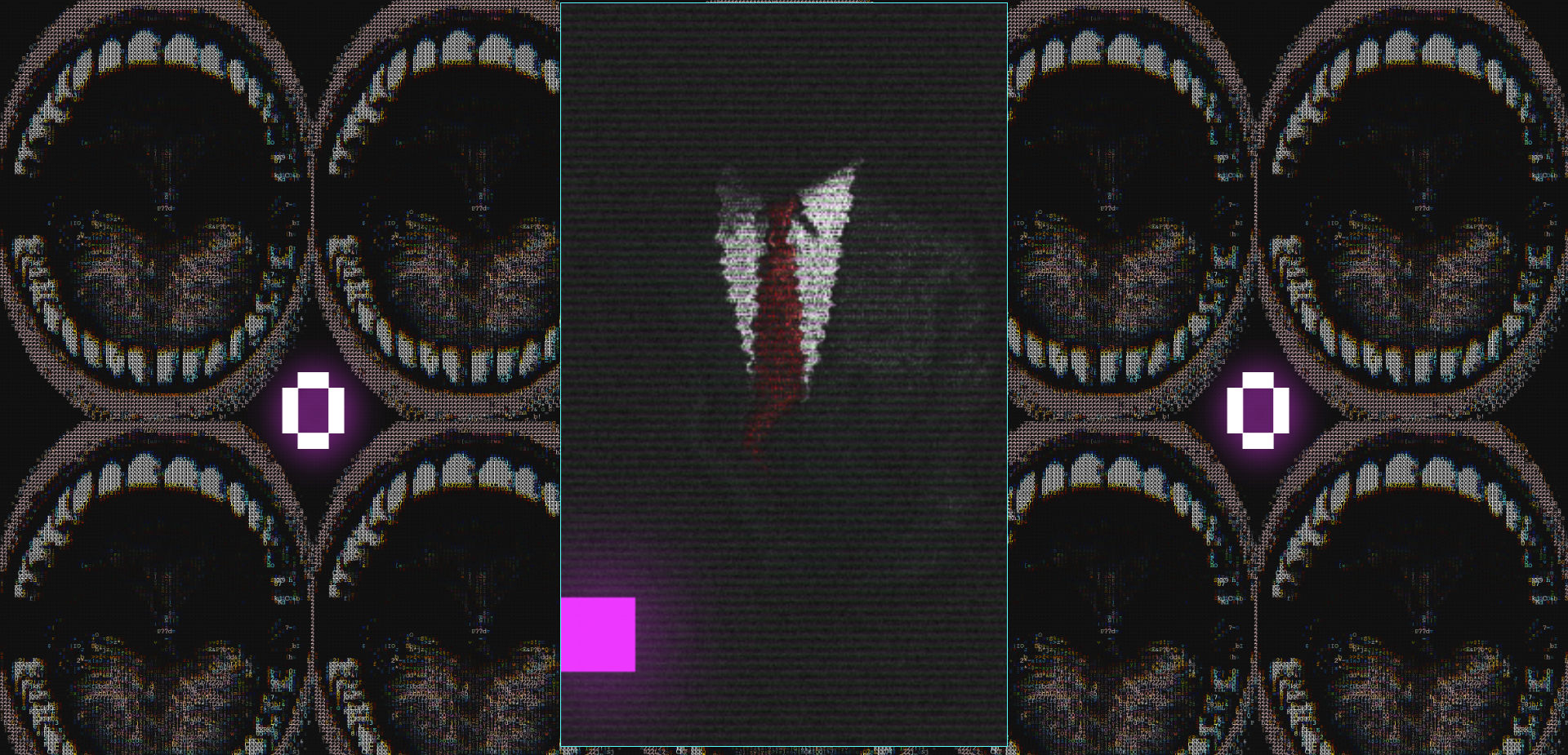
**LEVEL 2: Interview Questions**

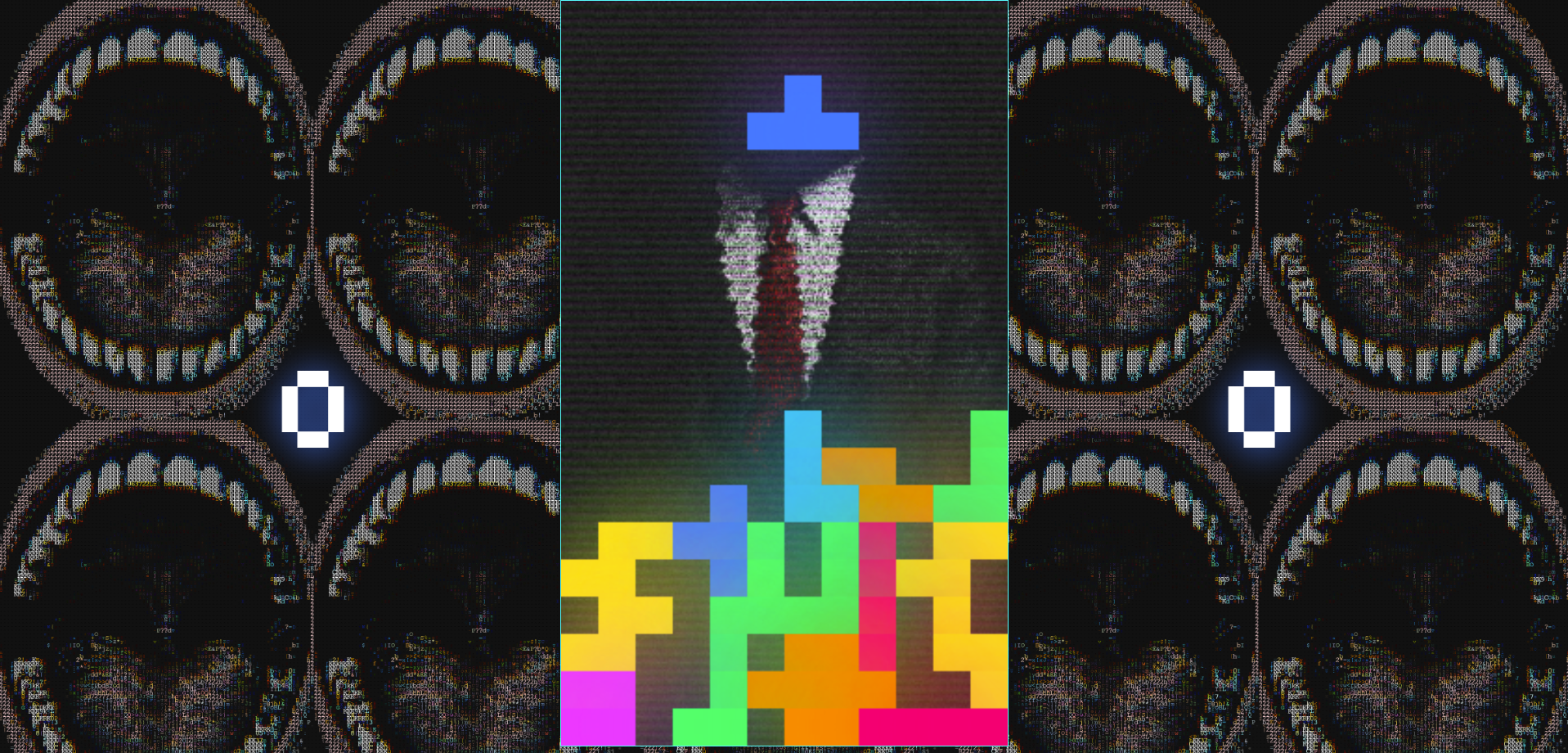
****

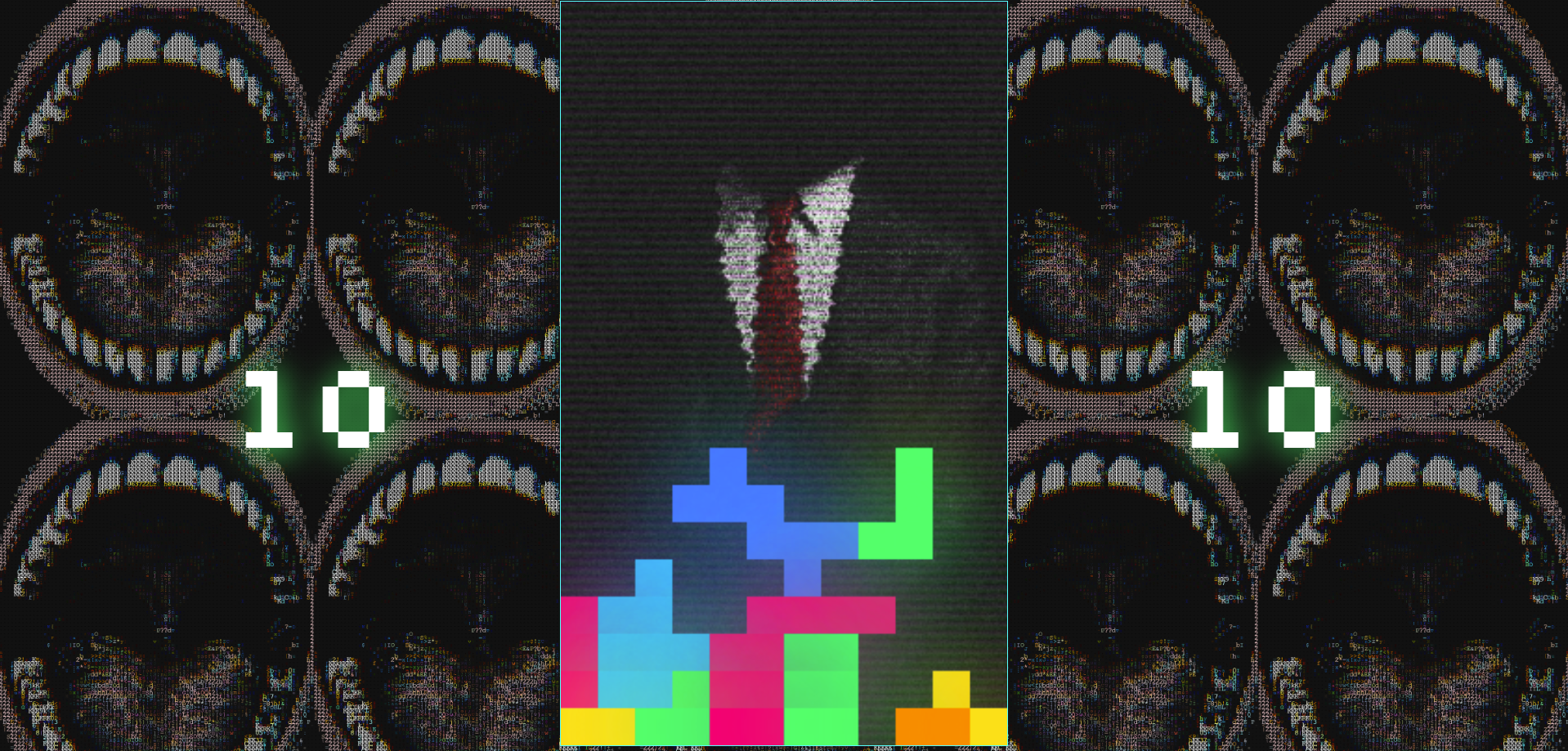
****

****

**LEVEL 3: Tetris**

****

****

****

<!DOCTYPE html>

<html lang="en">

  <head>

    <meta charset="UTF-8" />

    <link rel="icon" type="image/svg+xml" href="/vite.svg" />

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

    <title>Climbing The Corporate Ladder</title>

  </head>

  <body>

      <a href="" class="link" target="\_blank"></a>

      <canvas class="experience"></canvas>

    </div>

    <script type="module" src="/main.js"></script>

  </body>

</html>

import \* as THREE from "three"

import { BoxGeometry, Camera, MeshBasicMaterial } from "three";

import {GLTFLoader} from 'THREE/examples/jsm/loaders/GLTFLoader.js';

export default class Experience{

    constructor(canvas){

        this.canvas=canvas;

        const scene = new THREE.Scene();

        scene.background= new THREE.Color(0xCD8080);

        const camera = new THREE.PerspectiveCamera(39.6,window.innerWidth/window.innerHeight, 0.1,1500);

        camera.position.y=15;

        camera.position.z=27.5;

        camera.rotation.x-=Math.PI/3;

        const renderer = new THREE.WebGLRenderer();

        renderer.setSize(window.innerWidth, window.innerHeight);

        renderer.outputEncoding = THREE.sRGBEncoding;

        renderer.shadowMap.enabled = true;

        renderer.shadowMap.type = THREE.PCFSoftShadowMap;

        renderer.physicallyCorrectLights=true;

        renderer.toneMapping=THREE.ACESFilmicToneMapping;

        document.body.appendChild(renderer.domElement);

        const loader = new GLTFLoader();

        var cube;

        var pc1;

        var pc2;

        var pc3;

        loader.load(

            './gltfs/test\_optimised.glb',

            function(gltf){

                cube= gltf.scene;

                gltf.scene.traverse( function( node ) {

                    if ( node.isMesh ) { node.castShadow = true; node.receiveShadow=true; }

                } );

                scene.add(gltf.scene);

            }

        );

        loader.load(

            './gltfs/lvl1.glb',

            function(gltf){

                pc1= gltf.scene;

                gltf.scene.traverse( function( node ) {

                    if ( node.isMesh ) { node.castShadow = true; node.receiveShadow=true; }

                } );

                scene.add(gltf.scene);

            }

        );

        loader.load(

            './gltfs/lvl2.glb',

            function(gltf){

                pc2= gltf.scene;

                gltf.scene.traverse( function( node ) {

                    if ( node.isMesh ) { node.castShadow = true; node.receiveShadow=true; }

                } );

                scene.add(gltf.scene);

            }

        );

        loader.load(

            './gltfs/lvl3.glb',

            function(gltf){

                pc3= gltf.scene;

                gltf.scene.traverse( function( node ) {

                    if ( node.isMesh ) { node.castShadow = true; node.receiveShadow=true; }

                } );

                scene.add(gltf.scene);

            }

        );

        const amlight = new THREE.AmbientLight( 0xCD8080 );

        scene.add( amlight );

        const light1= new THREE.PointLight(0xffffff,122,50,2);

        light1.castShadow=true;

        light1.shadow.bias=-0.01;

        light1.shadow.radius=8;

        scene.add(light1);

        light1.position.x=0;

        light1.position.y=10;

        light1.position.z=-11;

        light1.shadow.mapSize.height= 1024;

        light1.shadow.mapSize.width= 1024;

        const light2= new THREE.PointLight(0xffffff,122,50,2);

        light2.castShadow=true;

        light2.shadow.bias=-0.01;

        light2.shadow.radius=8;

        scene.add(light2);

        light2.position.x=0;

        light2.position.y=10;

        light2.position.z=-4;

        light2.shadow.mapSize.height= 1024;

        light2.shadow.mapSize.width= 1024;

        const light3= new THREE.PointLight(0xffffff,122,50,2);

        light3.castShadow=true;

        light3.shadow.bias=-0.01;

        light3.shadow.radius=8;

        scene.add(light3);

        light3.position.x=0;

        light3.position.y=10;

        light3.position.z=3;

        light3.shadow.mapSize.height= 1024;

        light3.shadow.mapSize.width= 1024;

        const light4= new THREE.PointLight(0xffffff,122,50,2);

        light4.castShadow=true;

        light4.shadow.bias=-0.01;

        light4.shadow.radius=8;

        scene.add(light4);

        light4.position.x=0;

        light4.position.y=10;

        light4.position.z=10;

        light4.shadow.mapSize.height= 1024;

        light4.shadow.mapSize.width= 1024;

        const light5= new THREE.PointLight(0xffffff,122,50,2);

        light5.castShadow=true;

        light5.shadow.bias=-0.01;

        light5.shadow.radius=8;

        scene.add(light5);

        light5.position.x=0;

        light5.position.y=10;

        light5.position.z=18;

        light5.shadow.mapSize.height= 1024;

        light5.shadow.mapSize.width= 1024;

        const light6= new THREE.PointLight(0xffffff,122,50,2);

        light6.castShadow=true;

        light6.shadow.bias=-0.01;

        light6.shadow.radius=8;

        scene.add(light6);

        light5.position.x=0;

        light5.position.y=10;

        light6.position.z=-25;

        light6.shadow.mapSize.height= 1024;

        light6.shadow.mapSize.width= 1024;

        const link=document.querySelector('.link');

        function animate(){

            setTimeout( function() {

                requestAnimationFrame( animate );

            }, 1000 / 30 );

            document.onkeydown = checkKey;

            const zloc= camera.position.z;

            console.log(zloc);

            if(zloc>=25.5){

                link.innerHTML="";

                link.setAttribute("href","");

            }

            if(zloc<25.5 && zloc>=20.5){

                link.innerHTML="Dhyaan and Hrishik Present"

                link.setAttribute("href","");

            }

            if(zloc<20.5 && zloc>11.5){

                link.innerHTML="Climbing The Corporate Ladder";

            }

            if (zloc<=11.5 && zloc>=7.5){

                link.innerHTML="Play Level 1";

                link.setAttribute("href","Level-1/level-1.html");

                //link this to level 1

            }

            if (zloc<=3.5 && zloc>=2.5){

                link.innerHTML="Play Level 2";

                link.setAttribute("href","Level-2/level-2.html");

            }

            if (zloc<=-3.5){

                link.innerHTML="Play Level 3";

                link.setAttribute("href","Level-3/level-3.html");

            }

            }

            function checkKey(e) {

            e = e || window.event;

            if (e.keyCode == '38'&&camera.position.z>-8.4) {

            // up arrow

            camera.position.z-=0.1;

            }

         else if (e.keyCode == '40' && camera.position.z<29) {

            // down arrow

            camera.position.z+=0.08;

        }

            renderer.render(scene,camera);

        }

        animate();

    }

}

@import url('https://fonts.googleapis.com/css2?family=Press+Start+2P&family=Silkscreen&display=swap');

\*{

  margin: 0;

  padding: 0;

  box-sizing: border-box;

}

body{

  background-color: #CD8080;

}

.experience{

  position: fixed;

  height: 100vh;

  width: 100vw;

  z-index: -4;

}

.experience-canvas{

  height: 100%;

  width: 100%;

  z-index: -4;

}

.link{

  position: absolute;

  top: 50%;

    transform: translateY(-50%);

    left: 50%;

    transform: translateX(-50%);

  font-size: 5vh;

  font-family: silkscreen;

  text-decoration: none;

  text-align: center;

  color: #fff;

  text-shadow:

    0 0 42px #33FF33,

    0 0 82px #33FF33;

    transition: 0.1s;

}

.link:hover{

  text-shadow:

    0 0 42px #FFCC00,

    0 0 82px #FFCC00;

}

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <link rel="stylesheet" href="level-1-style.css">

    <title>Level 1</title>

</head>

<body id="body">

<div class="covering">

    <div class='deck'>

        <div class="card" data-name="linkedin">

            <div class="front">?</div>

            <img class="back" src="corpo logos\pix1.png">

        </div>

        <div class="card" data-name="linkedin">

            <div class="front">?</div>

            <img class="back" src="corpo logos\pix1.png">

        </div>

        <div class="card" data-name="netflix">

            <div class="front">?</div>

            <img class="back" src="corpo logos\pix2.png">

        </div>

        <div class="card" data-name="netflix">

            <div class="front">?</div>

            <img class="back" src="corpo logos\pix2.png">

        </div>

        <div class="card" data-name="tiktok">

            <div class="front">?</div>

            <img class="back" src="corpo logos\pix3.png">

        </div>

        <div class="card" data-name="tiktok">

            <div class="front">?</div>

            <img class="back" src="corpo logos\pix3.png">

        </div>

        <div class="card" data-name="twitter">

            <div class="front">?</div>

            <img class="back" src="corpo logos\pix4.png">

        </div>

        <div class="card" data-name="twitter">

            <div class="front">?</div>

            <img class="back" src="corpo logos\pix4.png">

        </div>

        <div class="card" data-name="instagram">

            <div class="front">?</div>

            <img class="back" src="corpo logos\pix5.png">

        </div>

        <div class="card" data-name="instagram">

            <div class="front">?</div>

            <img class="back" src="corpo logos\pix5.png">

        </div>

        <div class="card" data-name="whatsapp">

            <div class="front">?</div>

            <img class="back" src="corpo logos\pix6.png">

        </div>

        <div class="card" data-name="whatsapp">

            <div class="front">?</div>

            <img class="back" src="corpo logos\pix6.png">

        </div>

        <div class="card" data-name="facebook">

            <div class="front">?</div>

            <img class="back" src="corpo logos\pix7.png">

        </div>

        <div class="card"  data-name="facebook">

            <div class="front">?</div>

            <img class="back" src="corpo logos\pix7.png">

        </div>

        <div class="card" data-name="youtube">

            <div class="front">?</div>

            <img class="back" src="corpo logos\pix8.png">

        </div>

        <div class="card" data-name="youtube">

            <div class="front">?</div>

            <img class="back" src="corpo logos\pix8.png">

        </div>

    </div>

    <div class="details">

        <p class="time">Time: <span class="timer"><b>60</b></span>s</p>

      </div>

    <div class="details">

        <p class="score">Score: <span><b>2524140</b></span></p>

    </div>

</div>

    <script src="script.js"></script>

</body>

</html>

@import url('https://fonts.googleapis.com/css2?family=Press+Start+2P&family=Silkscreen&display=swap%27');

\*{

    margin: 0;

    padding: 0;

    box-sizing: border-box;

}

body{

    height: 100vh;

    display: flex;

    background-image: url('level-1-bg.png');

}

.covering{

    margin: auto;

}

.deck{

    height: 720px;

    width: 720px;

    display: flex;

    flex-wrap: wrap;

}

.card.shake{

    animation: shake 0.35s ease-in-out;

  }

  @keyframes shake {

    0%, 100%{

      transform: rotateZ(0deg);

    }

    20%{

      transform: rotateZ(4deg);

    }

    40%{

      transform: rotateZ(-4deg);

    }

    80%{

      transform: rotateZ(4deg);

    }

  }

.card{

    margin: 10px;

    height:calc(25% - 20px);

    width: calc(25% - 20px);

    position: relative;

    transform: scale(1);

    transition: transform 0.2s;

    transform-style: preserve-3d;

}

.card:active{

    transform: scale(0.95);

    transition: transform 0.5s;

}

.card.flip .back{

  transform: rotateY(0);

}

.card.flip .front{

  transform: rotateY(180deg);

}

.front,.back{

    height: 100%;

    width: 100%;

    position: absolute;

    border-radius: 5px;

    box-shadow: 5px 5px 0px black;

    backface-visibility: hidden;

    border: 4px solid black;

}

.back{

    object-fit: cover;

    z-index: -1;

    display: flex;

    transform: rotateY(180deg);

}

.front{

    font-size: auto;

    color:  #37E2D5;

    text-shadow: 2px 2px 0px black;

    background-color: #FBCB0A;

    line-height: 160px;

    text-align: center;

    font-size: 50px;

    font-family: 'Press Start 2P';

}

p{

  font-size: 2px;

  display: flex;

  align-items: center;

  justify-content: center;

}

.details{

  font-family: 'silkscreen';

  color: white;

  border: 3px solid white;

  width: 100%;

  margin-top: 5px;

  border-radius: 7px;

  background-image: url('pxArt.png');

  height: calc(100% / 4 - 30px);

  justify-content: space-between;

}

.timer{

  color: #EA047E

}

.details p{

  font-size: 45px;

  height: 85px;

}

.details p span{

  margin-left: 20px;

}

.details p b{

  font-weight: 500;

}

@media screen and (max-width: 530px) {

  .cards{

    height: 300px;

    width: 300px;

  }

  .card .back-view img{

    max-width: 35px;

  }

  .details{

    margin-top: 10px;

    padding: 0 15px;

    height: calc(100% / 4 - 20px);

  }

  .details p{

    height: 15px;

    font-size: 17px;

    padding-right: 13px;

  }

}

const cards = document.querySelectorAll('.card'),

timeTag = document.querySelector(".time b");

scoreTag = document.querySelector(".score b");

let hasFlippedCard = false;

let lockBoard = false;

let firstCard, secondCard;

let maxTime = 60;

let timeLeft = maxTime;

let timer;

let matchedCard = 0;

let isPlaying = false;

let score = 0;

function initTimer() {

    if(timeLeft <= 0) {

        lockBoard=true;

        const finScore=score;

        localStorage.setItem('score1',finScore);

        return clearInterval(timer);

    }

    timeLeft--;

    score = timeLeft\*42069

    timeTag.innerText = timeLeft;

    scoreTag.innerText = score;

    this.classList.add("score");

}

function flipCard(){

    if(!isPlaying) {

        isPlaying = true;

        timer = setInterval(initTimer, 1000);

    }

    if(lockBoard) return;

    if(this===firstCard) return;

    this.classList.add('flip');

    if(!hasFlippedCard){

        hasFlippedCard=true;

        firstCard=this;

    }

    else{

        hasFlippedCard=false;

        secondCard=this;

        checkForMatch();

    }

}

function checkForMatch(){

    let isMatch=firstCard.dataset.name === secondCard.dataset.name

    isMatch?disableCards():unFlipCard()

}

function disableCards(){

    firstCard.removeEventListener('click', flipCard);

    secondCard.removeEventListener('click', flipCard);

    matchedCard++;

    if(matchedCard == 8 && timeLeft > 0) {

        return clearInterval(timer);

    }

    resetBoard();

}

function unFlipCard(){

    firstCard.classList.add('shake')

    secondCard.classList.add('shake')

    lockBoard=true;

    setTimeout(()=>{

    firstCard.classList.remove('shake')

    secondCard.classList.remove('shake')

    firstCard.classList.remove('flip')

    secondCard.classList.remove('flip')

    resetBoard();

    }, 750)

}

function resetBoard()

{

    [hasFlippedCard, lockBoard] = [false,false];

    [firstCard, secondCard] = [ null, null];

}

(function shuffle(){

    cards.forEach(card=>{

        card.style.order=Math.floor((Math.random()\*16));

    })

})();

cards.forEach(card => card.addEventListener('click',flipCard))

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta http-equiv="X-UA-Compatible" content="IE=edge">

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

    <link rel="stylesheet" type="text/css" href="https://unpkg.com/augmented-ui@2/augmented-ui.min.css">

    <link rel="stylesheet" href="pstyle.css">

    <title>Level 2</title>

</head>

<body>

    <img src="pimg/Level text.png" alt="Level 2 " class="title-card">

    <div class="sticky"><img src="pimg/score  text.png" class="score-image" alt="score image"><p id="score"></p></div>

    <div class="alert" id="qbox" data-augmented-ui="tl-clip-y br-2-clip-y inlay">

        <h1 id="question"></h1>

        <span id="radiobtn">

        <input type="radio" name="q" id="opt1" value=""><label for="opt1" id="opt1label"></label><br><br>

        <input type="radio" name="q" id="opt2" value=""><label for="opt2" id="opt2label"></label><br><br>

        <input type="radio" name="q" id="opt3" value=""><label for="opt3" id="opt3label"></label><br><br>

        </span>

        <span id="sliderspan">

            <input type="range" min="0" max="10000" name="q5" id="slider" class="slider" value="5000" oninput="this.nextElementSibling.value = this.value">

            <output style="background: transparent">5000</output>

        </span>

        <button class="submitbutton" id="subbt" onclick="">OK</button>

    </div>

    <table>

        <tr><td><div id="guy1" class="i1"></div></td>

            <td><div id="guy2" class="i2" ></div></td>

            <td><div id="guy3" class="i3" ></div></td>

            <td><div id="guy4" class="i4" ></div></td>

            <td><div id="guy5" class="i5" ></div></td>

        </tr>

    </table>

    <script src="pscript.js"></script>

</body>

</html>

let Question1\_answered = false

let Question2\_answered = false

let Question3\_answered = false

let Question4\_answered = false

let Question5\_answered = false

let correctAnswer1 = false // turn this to true when correct answer and ket there be magic

let correctAnswer2 = false

let correctAnswer3 = false

let correctAnswer4 = false

let correctAnswer5 = false

const g1 = document.getElementById("guy1")

const g2 = document.getElementById("guy2")

const g3 = document.getElementById("guy3")

const g4 = document.getElementById("guy4")

const g5 = document.getElementById("guy5")

window.addEventListener("scroll", function(){

    updateScore();

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

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

    const opt1label =document.getElementById('opt1label');

    const opt2label =document.getElementById('opt2label');

    const opt3label =document.getElementById('opt3label');

    const opt1 =document.getElementById('opt1');

    const opt2 =document.getElementById('opt2');

    const opt3 =document.getElementById('opt3');

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

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

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

    sliderspan.style.display='none';

    radiobtn.style.display='block';

    let offset = window.pageYOffset

    console.log(offset)

    if(offset>= 3500 && Question1\_answered==false){

        g1.style.backgroundImage = "url(pimg/interviewer-1-eyes.png)";

        //this.alert("Hello1")   // turn this into a QnA

        radiobtn.style.display='none';

        qbox.style.display='block';

        question.innerHTML="Hello Developer, it seems you've cleared level-1. <br> Impressive."

        subbt.setAttribute('onclick', 'checkanswer1();');

    }

    g1.style.backgroundPositionY = offset \* 0.905 + "px"

    if(offset>= 5500 && Question2\_answered==false){

        g2.style.backgroundImage = "url(pimg/interviewer-2-eyes.png)";

        question.innerHTML='Where do you see yourself in five years?';

        opt1label.innerHTML='I see myself earning a decent amount by working hard at whatever work is assigned to me, impressing my boss and surpassing expectations.';

        opt1.setAttribute('value','medium')

        opt2label.innerHTML='I see myself being part of a digital revolution helping out to change the world through this corporation.';

        opt2.setAttribute('value','good')

        opt3label.innerHTML='I see myself climbing the corporate ladder and hopefully earning my place among you.';

        opt3.setAttribute('value','bad')

        qbox.style.display='block';

        subbt.setAttribute('onclick', 'checkanswer2();')

    }

    g2.style.backgroundPositionY = offset \* 0.92 + "px"

    if(offset>= 7500 && Question3\_answered==false){

        g3.style.backgroundImage = "url(pimg/interviewer-3-eyes.png)";

        question.innerHTML='What is the price of your soul?';

        radiobtn.style.display='none';

        sliderspan.style.display='block';

        console.log(radiobtn.style.display)

        qbox.style.display='block';

        subbt.setAttribute('onclick', 'checkanswer3();')

    }

    g3.style.backgroundPositionY = offset \* 0.93 + "px"

    if(offset>=6500 && Question4\_answered==false){

        g4.style.backgroundImage = "url(pimg/interviewer-4-eyes.png)";

        question.innerHTML='How do you view the privacy of your users?';

        opt1label.innerHTML='We must build privacy tools that put the user in control.￲';

        opt1.setAttribute('value','bad')

        opt2label.innerHTML='The quality of our services depends greatly on the data we collect. The users will understand.';

        opt2.setAttribute('value','medium')

        opt3label.innerHTML='User data keeps the software free. They won&lsquo;t admit it, but convenience is more important to the user than their own privacy';

        opt3.setAttribute('value','good')

        qbox.style.display='block';

        subbt.setAttribute('onclick', 'checkanswer4();')

    }

    g4.style.backgroundPositionY = offset \* 0.92 + "px"

    if(offset>=4500 && Question5\_answered==false){

        g5.style.backgroundImage = "url(pimg/interviewer-5-eyes.png)";

        radiobtn.style.display='none';

        qbox.style.display='block';

        question.innerHTML="Before we can send you off to him, we have three questions. Don't worry, it's just a formality."

        subbt.setAttribute('onclick', 'checkanswer5();');

    }

    g5.style.backgroundPositionY = offset \* 0.905 + "px"

    })

function updateScore() {

    document.getElementById('score').innerText = player.score;

}

const player = {

    score: localStorage.getItem('score1'),

};

function checkanswer1(){

    qbox.style.display='none';

    radiobtn.style.display='block';

    Question1\_answered = true;

}

function checkanswer2(){

    Question2\_answered = true;

    qbox.style.display='none'

    switch(document.querySelector('input[name="q"]:checked').value){

        case 'good':

            player.score+=10000

        case 'medium':

            player.score+=7000

        case 'bad':

            player.score+=5000

    }

    updateScore();

}

function checkanswer3(){

    Question3\_answered = true;

    qbox.style.display='none';

    if (document.getElementById('slider').value) {

        player.score -= document.getElementById('slider').value;

        updateScore();

    }

    radiobtn.style.display='block';

    sliderspan.style.display='none';

}

function checkanswer4(){

    Question4\_answered = true

    qbox.style.display='none'

    switch(document.querySelector('input[name="q"]:checked').value){

        case 'good':

            player.score+=10000

        case 'medium':

            player.score+=7000

        case 'bad':

            player.score+=5000

    }

    updateScore();

}

function checkanswer5(){

    qbox.style.display='none';

    radiobtn.style.display='block';

    Question5\_answered = true;

}

player.score=0;

updateScore();

@import url('https://fonts.googleapis.com/css2?family=Staatliches&family=Orbitron:wght@500&family=Press+Start+2P&family=Silkscreen&display=swap');

\*{

    border: 0px;

    padding: 0;

    margin: 0;

    background-color: #FE4646;

    background-size: cover;

    align-items: center;

    justify-content: center;

}

body{

    overflow-x: hidden;

}

.title-card{

    width: 100%;

    height: calc( 25vh \* 2);

    object-fit: cover;

}

.i1{

    background-image: url("pimg/interviewer-1.png");

    background-size:cover;

    background-size: contain;

    background-repeat: no-repeat;

    width: 100%;

    height: 8000px;

    object-fit: cover;

    object-position: top;

    margin-top: 100px;

    transition: background-image 0.1s;

}

.i2{

    background-image: url("pimg/interviewer-2.png");

    background-size:cover;

    background-size: contain;

    background-repeat: no-repeat;

    width: 100%;

    height: 8000px;

    object-fit: cover;

    object-position: top;

    margin-top: 98px;

    transition: background-image 0.1s;

}

.i3{

    background-image: url("pimg/interviewer-3.png");

    background-size:cover;

    background-size: contain;

    background-repeat: no-repeat;

    width: 100%;

    height: 8000px;

    object-fit: cover;

    object-position: top;

    margin-top: 94px;

    transition: background-image 0.1s;

}

.i4{

    background-image: url("pimg/interviewer-4.png");

    background-size:cover;

    background-size: contain;

    background-repeat: no-repeat;

    width: 100%;

    height: 8000px;

    object-fit: cover;

    object-position: top;

    margin-top: 98px;

    transition: background-image 0.1s;

}

.i5{

    background-image: url("pimg/interviewer-5.png");

    background-size:cover;

    background-size: contain;

    background-repeat: no-repeat;

    width: 100%;

    height: 8000px;

    object-fit: cover;

    object-position: top;

    margin-top: 100px;

    transition: background-image 0.1s;

}

table{

    z-index: 2;

    align-items: center;

    justify-content: center;

    width:99vw;

    border-spacing: 0px;

}

div.sticky {

    position: sticky;

    margin: auto;

    width: 30vw;

    top: 0;

    /\*background-color: #FE4646;

    text-align:left;

    font-family: 'Press Start 2P';

    font-size: 70px;\*/

    color: #ffa5f6;

    z-index: 0;

    /\*text-shadow:

        0 0 30px #ffa6f8

        0 0 30px rgb(38, 0, 255);\*/

    }

.score-image{

    width: 30vw;

    margin: auto;

    mix-blend-mode: lighten;

    height: 7vw;

    object-fit: cover;

}

#score{

    width: 30vw;

    margin: auto;

    text-align: center;

    font-size: 7vh;

    font-family: orbitron;

}

.alert{

    /\*augmented ui\*/

    --aug-inlay-bg: rgb(35, 26, 26);

    --aug-tl: 17px;

    --aug-br:13px;

    /\*vanilla css\*/

    font-size: 12px;

    display: none;

    background: linear-gradient(55deg, #d7d7d7,#ffffff);

    font-family: 'Silkscreen';

    align-items: center;

    color: black;

    z-index: 1;

    top: 57%;

    transform: translateY(-43%);

    left: 50%;

    transform: translateX(-50%);

    position: fixed;

    text-align: left;

    padding-left: 100px;

    padding-right: 100px;

    padding-top: 50px;

    padding-bottom: 50px;

}

label{

    background-color: transparent;

}

.submitbutton{

    font-size: 20px;

    line-height: 30px;

    padding-left: 20px;

    padding-right: 20px;

    border: 1px solid #855454;

    border-radius:50px;

    box-shadow: 3px 3px 0px #855454;

    transition: 0.1s;

}

.submitbutton:hover{

    opacity: 0.8;

}

.submitbutton:active{

    margin-left: 3px;

    margin-top: 3px;

    box-shadow: none;

}

#radiobtn{

    background: transparent;

    color: white;

}

.slider{

    width: 90%;

    background-color: transparent;

}

#sliderspan{

    background-color: transparent;

}

output{

    color: white;

}

h1{

    color: white;

    background-color: transparent;

}

@import url('https://fonts.googleapis.com/css2?family=Staatliches&family=Orbitron:wght@500&family=Press+Start+2P&family=Silkscreen&display=swap');

\*{

    border: 0px;

    padding: 0;

    margin: 0;

    background-color: #FE4646;

    background-size: cover;

    align-items: center;

    justify-content: center;

}

body{

    overflow-x: hidden;

}

.title-card{

    width: 100%;

    height: calc( 25vh \* 2);

    object-fit: cover;

}

.i1{

    background-image: url("pimg/interviewer-1.png");

    background-size:cover;

    background-size: contain;

    background-repeat: no-repeat;

    width: 100%;

    height: 8000px;

    object-fit: cover;

    object-position: top;

    margin-top: 100px;

    transition: background-image 0.1s;

}

.i2{

    background-image: url("pimg/interviewer-2.png");

    background-size:cover;

    background-size: contain;

    background-repeat: no-repeat;

    width: 100%;

    height: 8000px;

    object-fit: cover;

    object-position: top;

    margin-top: 98px;

    transition: background-image 0.1s;

}

.i3{

    background-image: url("pimg/interviewer-3.png");

    background-size:cover;

    background-size: contain;

    background-repeat: no-repeat;

    width: 100%;

    height: 8000px;

    object-fit: cover;

    object-position: top;

    margin-top: 94px;

    transition: background-image 0.1s;

}

.i4{

    background-image: url("pimg/interviewer-4.png");

    background-size:cover;

    background-size: contain;

    background-repeat: no-repeat;

    width: 100%;

    height: 8000px;

    object-fit: cover;

    object-position: top;

    margin-top: 98px;

    transition: background-image 0.1s;

}

.i5{

    background-image: url("pimg/interviewer-5.png");

    background-size:cover;

    background-size: contain;

    background-repeat: no-repeat;

    width: 100%;

    height: 8000px;

    object-fit: cover;

    object-position: top;

    margin-top: 100px;

    transition: background-image 0.1s;

}

table{

    z-index: 2;

    align-items: center;

    justify-content: center;

    width:99vw;

    border-spacing: 0px;

}

div.sticky {

    position: sticky;

    margin: auto;

    width: 30vw;

    top: 0;

    /\*background-color: #FE4646;

    text-align:left;

    font-family: 'Press Start 2P';

    font-size: 70px;\*/

    color: #ffa5f6;

    z-index: 0;

    /\*text-shadow:

        0 0 30px #ffa6f8

        0 0 30px rgb(38, 0, 255);\*/

    }

.score-image{

    width: 30vw;

    margin: auto;

    mix-blend-mode: lighten;

    height: 7vw;

    object-fit: cover;

}

#score{

    width: 30vw;

    margin: auto;

    text-align: center;

    font-size: 7vh;

    font-family: orbitron;

}

.alert{

    /\*augmented ui\*/

    --aug-inlay-bg: rgb(35, 26, 26);

    --aug-tl: 17px;

    --aug-br:13px;

    /\*vanilla css\*/

    font-size: 12px;

    display: none;

    background: linear-gradient(55deg, #d7d7d7,#ffffff);

    font-family: 'Silkscreen';

    align-items: center;

    color: black;

    z-index: 1;

    top: 57%;

    transform: translateY(-43%);

    left: 50%;

    transform: translateX(-50%);

    position: fixed;

    text-align: left;

    padding-left: 100px;

    padding-right: 100px;

    padding-top: 50px;

    padding-bottom: 50px;

}

label{

    background-color: transparent;

}

.submitbutton{

    font-size: 20px;

    line-height: 30px;

    padding-left: 20px;

    padding-right: 20px;

    border: 1px solid #855454;

    border-radius:50px;

    box-shadow: 3px 3px 0px #855454;

    transition: 0.1s;

}

.submitbutton:hover{

    opacity: 0.8;

}

.submitbutton:active{

    margin-left: 3px;

    margin-top: 3px;

    box-shadow: none;

}

#radiobtn{

    background: transparent;

    color: white;

}

.slider{

    width: 90%;

    background-color: transparent;

}

#sliderspan{

    background-color: transparent;

}

output{

    color: white;

}

h1{

    color: white;

    background-color: transparent;

}

@import url('https://fonts.googleapis.com/css2?family=Orbitron:wght@500&family=Press+Start+2P&family=Silkscreen&display=swap');

body {

    margin: 0;

    padding: 0;

    text-align: center;

    background-image: url('immmg/asciieffectmouth.png');

    background-size: 385px;

  }

  canvas {

    top: 5px;

    border:  1px solid cyan;

    background-image: url("immmg/suit background.png");

    height: 98%;

    margin: 0;

    position: relative;

  }

 #score2

  { font-family: "Silkscreen";

    font-size: 150px;

    border-radius: 7px;

    height: 34px;

    margin: 10;

    width: 200px;

    z-index: 1;

    line-height: 4vh;

    top: 50vh;

    position: fixed;

    left: 74.5%;

    text-align: center;

  }

  #score1

  { font-family: "Silkscreen";

    font-size: 150px;

    border-radius: 7px;

    height: 34px;

    margin: 10;

    width: 200px;

    z-index: 1;

    line-height: 4vh;

    top: 50vh;

    position: fixed;

    left: 14.25%;

    text-align: center;

  }

const canvas = document.getElementById('tetris');

const context = canvas.getContext('2d');

let color = document.body.style.color;

context.scale(20, 20);

function arenaSweep() {

    let rowCount = 1;

    outer: for (let y = arena.length -1; y > 0; --y) {

        for (let x = 0; x < arena[y].length; ++x) {

            if (arena[y][x] === 0) {

                continue outer;

            }

        }

        const row = arena.splice(y, 1)[0].fill(0);

        arena.unshift(row);

        ++y;

        player.score += rowCount \* 10;

        rowCount \*= 2;

    }

}

function collide(arena, player) {

    const m = player.matrix;

    const o = player.pos;

    for (let y = 0; y < m.length; ++y) {

        for (let x = 0; x < m[y].length; ++x) {

            if (m[y][x] !== 0 &&

               (arena[y + o.y] &&

                arena[y + o.y][x + o.x]) !== 0) {

                return true;

            }

        }

    }

    return false;

}

function createMatrix(w, h) {

    const matrix = [];

    while (h--) {

        matrix.push(new Array(w).fill(0));

    }

    return matrix;

}

function createPiece(type)

{

    if (type === 'I') {

        return [

            [0, 1, 0, 0],

            [0, 1, 0, 0],

            [0, 1, 0, 0],

            [0, 1, 0, 0],

        ];

    } else if (type === 'L') {

        return [

            [0, 2, 0],

            [0, 2, 0],

            [0, 2, 2],

        ];

    } else if (type === 'J') {

        return [

            [0, 3, 0],

            [0, 3, 0],

            [3, 3, 0],

        ];

    } else if (type === 'O') {

        return [

            [4, 4],

            [4, 4],

        ];

    } else if (type === 'Z') {

        return [

            [5, 5, 0],

            [0, 5, 5],

            [0, 0, 0],

        ];

    } else if (type === 'S') {

        return [

            [0, 6, 6],

            [6, 6, 0],

            [0, 0, 0],

        ];

    } else if (type === 'T') {

        return [

            [0, 7, 0],

            [7, 7, 7],

            [0, 0, 0],

        ];

    }

}

function drawMatrix(matrix, offset) {

    matrix.forEach((row, y) => {

        row.forEach((value, x) => {

            if (value !== 0) {

                context.fillStyle = colors[value];

                context.fillRect(x + offset.x,

                                 y + offset.y,

                                 1, 1);

                context.shadowColor = colors[value];//here

                context.shadowBlur = 55;///extent of the shadow

                context.lineJoin = "bevel";

                context.strokeRect(30, 0, 0, 0);

                document.body.style.color = "#FFF";

                document.body.style.textShadow = '3px 3px 42px ' + colors[value], '3px 3px 82px ' + colors[value];

            }

        });

    });

}

function draw() {

    const image = new Image(60, 45);

    image.onload = drawImageActualSize;

    image.src = "immmg/suit background.png";

    function drawImageActualSize() {

      context.drawImage(this, 0, 0, 12, 20);

    }

    drawMatrix(arena, {x: 0, y: 0});

    drawMatrix(player.matrix, player.pos);

}

function merge(arena, player) {

    player.matrix.forEach((row, y) => {

        row.forEach((value, x) => {

            if (value !== 0) {

                arena[y + player.pos.y][x + player.pos.x] = value;

            }

        });

    });

}

function rotate(matrix, dir) {

    for (let y = 0; y < matrix.length; ++y) {

        for (let x = 0; x < y; ++x) {

            [

                matrix[x][y],

                matrix[y][x],

            ] = [

                matrix[y][x],

                matrix[x][y],

            ];

        }

    }

    if (dir > 0) {

        matrix.forEach(row => row.reverse());

    } else {

        matrix.reverse();

    }

}

function playerDrop() {

    player.pos.y++;

    if (collide(arena, player)) {

        player.pos.y--;

        merge(arena, player);

        playerReset();

        arenaSweep();

        updateScore();

    }

    dropCounter = 0;

}

function playerMove(offset) {

    player.pos.x += offset;

    if (collide(arena, player)) {

        player.pos.x -= offset;

    }

}

function playerReset() {

    const pieces = 'TJLOSZI';

    player.matrix = createPiece(pieces[pieces.length \* Math.random() | 0]);

    player.pos.y = 0;

    player.pos.x = (arena[0].length / 2 | 0) -

                   (player.matrix[0].length / 2 | 0);

    if (collide(arena, player)) {

        arena.forEach(row => row.fill(0));

        player.score = 0;

        updateScore();

    }

}

function playerRotate(dir) {

    const pos = player.pos.x;

    let offset = 1;

    rotate(player.matrix, dir);

    while (collide(arena, player)) {

        player.pos.x += offset;

        offset = -(offset + (offset > 0 ? 1 : -1));

        if (offset > player.matrix[0].length) {

            rotate(player.matrix, -dir);

            player.pos.x = pos;

            return;

        }

    }

}

let dropCounter = 0;

let dropInterval = 350; //game speed

let lastTime = 0;

function update(time = 0) {

    const deltaTime = time - lastTime;

    dropCounter += deltaTime;

    if (dropCounter > dropInterval) {

        playerDrop();

    }

    lastTime = time;

    draw();

    requestAnimationFrame(update);

}

function updateScore() {

    document.getElementById('score1').innerText = player.score;

    document.getElementById('score2').innerText = player.score;

}

document.addEventListener('keydown', event => {

    if (event.keyCode === 37) {

        playerMove(-1);

    } else if (event.keyCode === 39) {

        playerMove(1);

    } else if (event.keyCode === 40) {

        playerDrop();

    } else if (event.keyCode === 32) {

        playerRotate(-1);

    } else if (event.keyCode === 87) {

        playerRotate(1);

    }

});

const colors = [

    null,

    '#FF0D72',

    '#0DC2FF',

    '#0DFF72',

    '#F538FF',

    '#FF8E0D',

    '#FFE138',

    '#3877FF',

];

const arena = createMatrix(12, 20);

const player = {

    pos: {x: 0, y: 0},

    matrix: null,

    score: 0,

};

playerReset();

updateScore();

update();