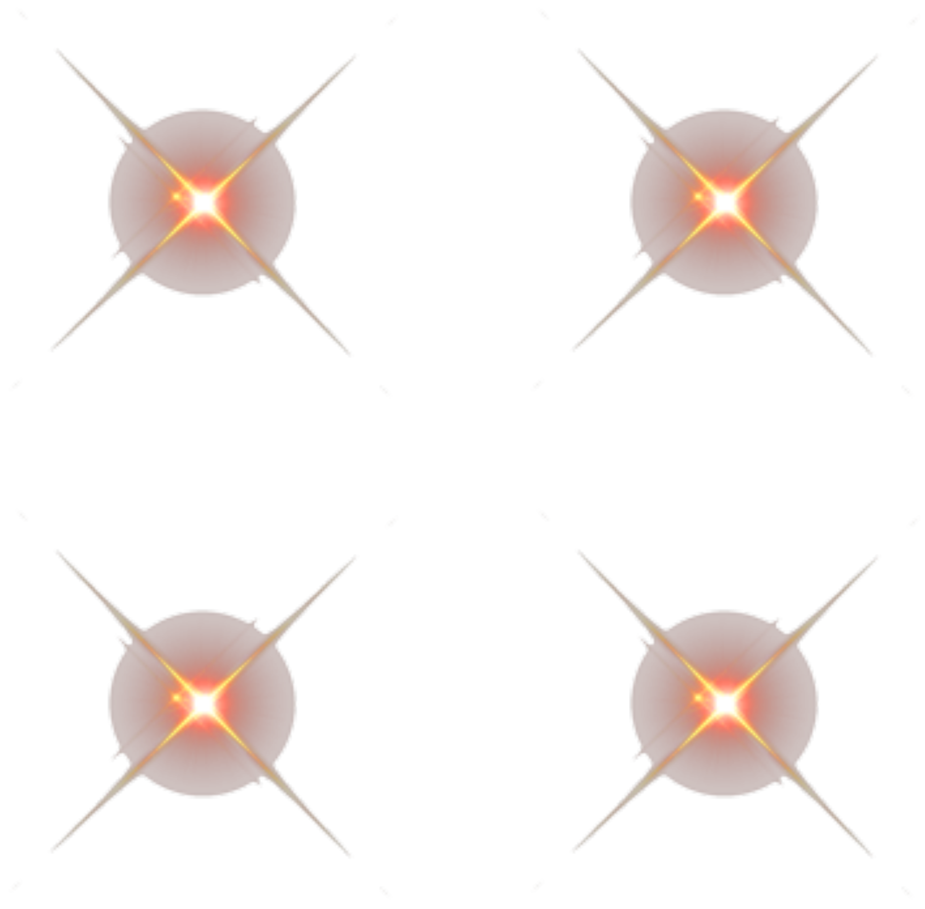
Tech@JesseJesse.com

GALAXY SPACE FORCE
CREATOR BLOB

https://github.com/
sudo-self/space



                      style.css
index.html   script.js

1. style.css

```
body {
    margin: 0;
    overflow: hidden;
    width: 100vw;
    height: 100vh;
    background-image: url(https://<any
space photo you choose goes here>);
    background-size: cover;
    backdrop-filter:  brightness(50%);
}

canvas {
    display: block;
}

#canvas_container {
    width: 100%;
    height: 100vh;
}
```

```
button {
    position: absolute;
    bottom: 5%;
    left: 50%;
    transform: translateX(-50%);
    border: 1px solid white;
    border-radius: 5px;
    font-size: 0.9rem;
    padding: 0.5rem 0.9em;
    background: #000000;
    color: white;
    -webkit-font-smoothing: antialiased;
    font-weight: bold;
    cursor: pointer;
    transition: all .3s;
}

button:hover {
    background: #ffffff;
    color: #000000;
}
```

2. index.html

```
<!DOCTYPE html>

<html lang="en" >
```

```
<head>
  <meta charset="UTF-8">
  <title>JEDI FORCE BLOB</title>
  <meta name="viewport"
content="width=device-width, initial-
scale=1"><link rel="stylesheet" href="./
style.css">

</head>
<body>

<div id="canvas_container"></div>



  <script src='https://cdn.jsdelivr.net/
npm/three@0.121.1/build/three.min.js'></
script>
```

```
<script src='https://cdn.jsdelivr.net/
npm/three@0.121.1/examples/js/controls/
OrbitControls.js'></script>
<script src='https://
cdnjs.cloudflare.com/ajax/libs/simplex-
noise/2.4.0/simplex-noise.min.js'></
script><script  src="./script.js"></
script>

</body>
</html>
```

3. script.js

```
let renderer,
scene,
camera,
sphereBg,
nucleus,
stars,
controls,
container =
document.getElementById("canvas_container
"),
timeout_Debounce,
noise = new SimplexNoise(),
cameraSpeed = 0,
blobScale = 3;



init();
animate();
```

```
function init() {
    scene = new THREE.Scene();

    camera = new
THREE.PerspectiveCamera(55,
window.innerWidth / window.innerHeight,
0.01, 1000)
    camera.position.set(0,0,230);

    const directionalLight = new
THREE.DirectionalLight("#fff", 2);
    directionalLight.position.set(0, 50,
-20);
    scene.add(directionalLight);

    let ambientLight = new
```

```
THREE.AmbientLight("#ffffff", 1);
    ambientLight.position.set(0, 20, 20);
    scene.add(ambientLight);

    renderer = new THREE.WebGLRenderer({
        antialias: true,
        alpha: true
    });

renderer.setSize(container.clientWidth,
container.clientHeight);

renderer.setPixelRatio(window.devicePixel
Ratio);

container.appendChild(renderer.domElement
);
```

```
    //OrbitControl
    controls = new
THREE.OrbitControls(camera,
renderer.domElement);
    controls.autoRotate = true;
    controls.autoRotateSpeed = 4;
    controls.maxDistance = 350;
    controls.minDistance = 150;
    controls.enablePan = false;

    const loader = new
THREE.TextureLoader();
    const textureSphereBg =
loader.load('https://i.ibb.co/4gHcRZD/
bg3-je3ddz.jpg');
    const texturenucleus =
loader.load('https://i.ibb.co/hcN2qXk/
star-nc8wkw.jpg');
    const textureStar =
loader.load("https://i.ibb.co/ZKsdYSz/p1-
g3zb2a.png");
    const texture1 =
loader.load("https://i.ibb.co/F8by6wW/p2-
b3gnym.png");
    const texture2 =
loader.load("https://i.ibb.co/yYS2yx5/p3-
ttfn70.png");
    const texture4 =
loader.load("https://i.ibb.co/yWfKkHh/p4-
avirap.png");


    /*  Nucleus  */
    texturenucleus.anisotropy = 16;
    let icosahedronGeometry = new
THREE.IcosahedronGeometry(30, 10);
    let lambertMaterial = new
THREE.MeshPhongMaterial({ map:
texturenucleus });
    nucleus = new
THREE.Mesh(icosahedronGeometry,
lambertMaterial);
    scene.add(nucleus);


    /*    Sphere  Background   */
    textureSphereBg.anisotropy = 16;
    let geometrySphereBg = new
THREE.SphereBufferGeometry(150, 40, 40);
    let materialSphereBg = new
```

```
THREE.MeshBasicMaterial({
        side: THREE.BackSide,
        map: textureSphereBg,
    });
    sphereBg = new
THREE.Mesh(geometrySphereBg,
materialSphereBg);
    scene.add(sphereBg);



    /*    Moving Stars   */
    let starsGeometry = new
THREE.Geometry();


    for (let i = 0; i < 50; i++) {
        let particleStar =
randomPointSphere(150);
```

```
        particleStar.velocity =
THREE.MathUtils.randInt(50, 200);

        particleStar.startX =
particleStar.x;
        particleStar.startY =
particleStar.y;
        particleStar.startZ =
particleStar.z;


starsGeometry.vertices.push(particleStar)
;
    }
    let starsMaterial = new
THREE.PointsMaterial({
        size: 5,
        color: "#ffffff",
        transparent: true,
        opacity: 0.8,
        map: textureStar,
        blending: THREE.AdditiveBlending,
    });
    starsMaterial.depthWrite = false;
    stars = new
THREE.Points(starsGeometry,
starsMaterial);
    scene.add(stars);



    /*    Fixed Stars   */
    function createStars(texture, size,
total) {
        let pointGeometry = new
THREE.Geometry();
        let pointMaterial = new
THREE.PointsMaterial({
            size: size,
            map: texture,
            blending:
THREE.AdditiveBlending,
        });

        for (let i = 0; i < total; i++) {
            let radius =
THREE.MathUtils.randInt(149, 70);
            let particles =
randomPointSphere(radius);

pointGeometry.vertices.push(particles);
```

```
        }
        return new
THREE.Points(pointGeometry,
pointMaterial);
    }
    scene.add(createStars(texture1, 15,
20));
    scene.add(createStars(texture2, 5,
5));
    scene.add(createStars(texture4, 7,
5));


    function randomPointSphere (radius) {
        let theta = 2 * Math.PI *
Math.random();
        let phi = Math.acos(2 *
Math.random() - 1);
        let dx = 0 + (radius *
Math.sin(phi) * Math.cos(theta));
        let dy = 0 + (radius *
Math.sin(phi) * Math.sin(theta));
        let dz = 0 + (radius *
Math.cos(phi));
        return new THREE.Vector3(dx, dy,
dz);
    }
}


function animate() {

    //Stars  Animation
```

```
stars.geometry.vertices.forEach(function
(v) {
        v.x += (0 - v.x) / v.velocity;
        v.y += (0 - v.y) / v.velocity;
        v.z += (0 - v.z) / v.velocity;

        v.velocity -= 0.3;

        if (v.x <= 5 && v.x >= -5 && v.z
<= 5 && v.z >= -5) {
            v.x = v.startX;
            v.y = v.startY;
            v.z = v.startZ;
            v.velocity =
THREE.MathUtils.randInt(50, 300);
        }
    });
    //Nucleus Animation

nucleus.geometry.vertices.forEach(functio
n (v) {
        let time = Date.now();
        v.normalize();
        let distance =
nucleus.geometry.parameters.radius +
noise.noise3D(
            v.x + time * 0.0005,
            v.y + time * 0.0003,
            v.z + time * 0.0008
        ) * blobScale;
        v.multiplyScalar(distance);
    })
    nucleus.geometry.verticesNeedUpdate =
true;
    nucleus.geometry.normalsNeedUpdate =
true;

nucleus.geometry.computeVertexNormals();

nucleus.geometry.computeFaceNormals();
    nucleus.rotation.y += 0.002;



    //Sphere Beckground Animation
    sphereBg.rotation.x += 0.002;
    sphereBg.rotation.y += 0.002;
    sphereBg.rotation.z += 0.002;
    controls.update();
    stars.geometry.verticesNeedUpdate =
true;
    renderer.render(scene, camera);
    requestAnimationFrame(animate);
}




/*     Resize     */
window.addEventListener("resize", () => {
    clearTimeout(timeout_Debounce);
    timeout_Debounce =
setTimeout(onWindowResize, 80);
});
function onWindowResize() {
    camera.aspect =
container.clientWidth /
container.clientHeight;
    camera.updateProjectionMatrix();

renderer.setSize(container.clientWidth,
container.clientHeight);
}



/*     Fullscreen btn     */
// let fullscreen;
// let fsEnter =
document.getElementById('fullscr');
// fsEnter.addEventListener('click',
function (e) {
//     e.preventDefault();
//      if (!fullscreen) {
//          fullscreen = true;
//
document.documentElement.requestFullscree
n();
//          fsEnter.innerHTML = "Exit
Fullscreen";
//      }
//      else {
//          fullscreen = false;
//          document.exitFullscreen();
//          fsEnter.innerHTML = "Go
Fullscreen";
//      }
// });
```