CODING ONE FINAL PROJECT
YIFAN GU -20020580-

Mimic link:
https://mimicproject.com/code/298d7d44-f05d-889f-fff8-765c36338abb
Youtube link:
https://www.youtube.com/watch?v=tpf5_iUmz-Y&feature=youtu.be

notes:

```
// Based on several ideas and shaders on Shadertoy:
   // 'Plasma Globe' by nimitz (https://www.shadertoy.com/view/XsjXRm)
   // 'Supernova remnant' by Duke (https://www.shadertoy.com/view/MdKXzc)
   // 'Creation by Silexars' by Danilo Guanabara (https://www.shadertoy.com/view/XsXXDn)


   // License Creative Commons Attribution-Noncommercial-ShareAlike 3.0 Unported
License

/////purpule cloud parameters
   const float nudge = 0.9; // random range (size of perpendicular vector)
   float normalizer = 1.0 / sqrt(1.0 + nudge*nudge);       // pythagorean theorem on that
perpendicular to
   const int spiralnoisec_num = 3; // cloud shape
   const float initial_iter = 5.0; // cloud density
   const float radius_inv = -5.0; // cloud radius (must be negative num)
   const int raymarch_loop_num    = 30;

//// iq's noise

float noise1( in vec3 x )
        {
          vec3 p = floor(x);
          vec3 f = fract(x);

// maintain scale
   float SpiralNoiseC(vec3 p)

// add sin and cos scaled inverse with the frequency
                n += -abs(sin(p.y*iter) + cos(p.x*iter)) / iter; // abs for a ridged look
                // rotate by adding perpendicular and scaling down
                p.xy += vec2(p.y, -p.x) * nudge;
                p.xy *= normalizer;
                // rotate on other axis
```

```
            p.xz += vec2(p.z, -p.x) * nudge;
            p.xz *= normalizer;
            // increase the frequency
            iter *= 1.733733;




// assign color to the media
   vec3 computeColor( float density, float radius )
   {
      // color based on density alone, gives impression of occlusion within the media
      vec3 result = mix( vec3(1.0,0.9,0.8), vec3(0.4,0.15,0.1), density );

      // color added to the media
      vec3 colCenter = 7.*vec3(0.8,1.0,1.0);
      vec3 colEdge = 1.5*vec3(0.48,0.53,0.5);
      result *= mix( colCenter, colEdge, min( (radius+.05)/.9, 1.15 ) );

      return result;


      // Ray/Sphere intersect function to simulate supernova
         // from this link: https://www.shadertoy.com/view/3lVyRh
         bool RaySphereIntersect(vec3 org, vec3 dir, out float near, out float far)

      // Applies the filmic curve from John Hable's presentation
         vec3 ToneMapFilmicALU(vec3 _color)

      // ro: ray origin
            // rd: direction of the ray
              vec3 rd = normalize(vec3((fragCoord.xy-0.5*iResolution.xy)/iResolution.y, 1.5));
            vec3 ro = vec3(0.0, 0.1, radius_inv+key*0.6);

      // ld, td: local, total density
            // w: weighting factor
            float ld=0., td=0., w=0.;

            // t: length of the ray
            // d: distance function
            float d=1., t=0.;

      // Loop break conditions.
```

```
            if(td>0.9 || d<0.1*t || t>10. || sum.a > 0.99 || t>max_dist) break;

                // evaluate distance function
                float d = map1(pos);

            // change this string to control density
            d = max(d,0.0);

                // point light calculations
                vec3 ldst = vec3(0.0)-pos;
                float lDist = max(length(ldst), 0.001);

                // the color of light
                vec3 lightColor=vec3(1.0,0.5,1.25);

    // Based on 'Creation by Silexars' by Danilo Guanabara
(https://www.shadertoy.com/view/XsXXDn)
       void mainImage2(out vec4 fragColor, in vec2 fragCoord )

    // camera
        vec3 ro = vec3(0.,0.,5.);

        // zoom the globe with average audio frequency
        vec3 rd = normalize(vec3(p*.7, pow(AverageFrequency/150.0, 2.4) * -1.5));

    //load audio
    const audioListener = new THREE.AudioListener();
        audio = new THREE.Audio(audioListener);

        const audioLoader = new THREE.AudioLoader();
        audioLoader.load('sunny.mp3', (buffer) => {
          audio.setBuffer(buffer);
          audio.setLoop(true);
    y();
        });
    // audio analyser: get the average frequency of the sound
        const fftSize = 32;
        analyser = new THREE.AudioAnalyser(audio, fftSize);

    window.addEventListener("click", function() {
          if (!audio.isPlaying)
            audio.play();
        });
```

```
// track mouse move
var imouse = new THREE.Vector2();
      window.addEventListener("mousemove", function(evt) {
         imouse.x = evt.clientX;
         imouse.y = evt.clientY;
      })

// all the notes is in mimic, this is mostly I used ,and I 'll show the process of the image.
```