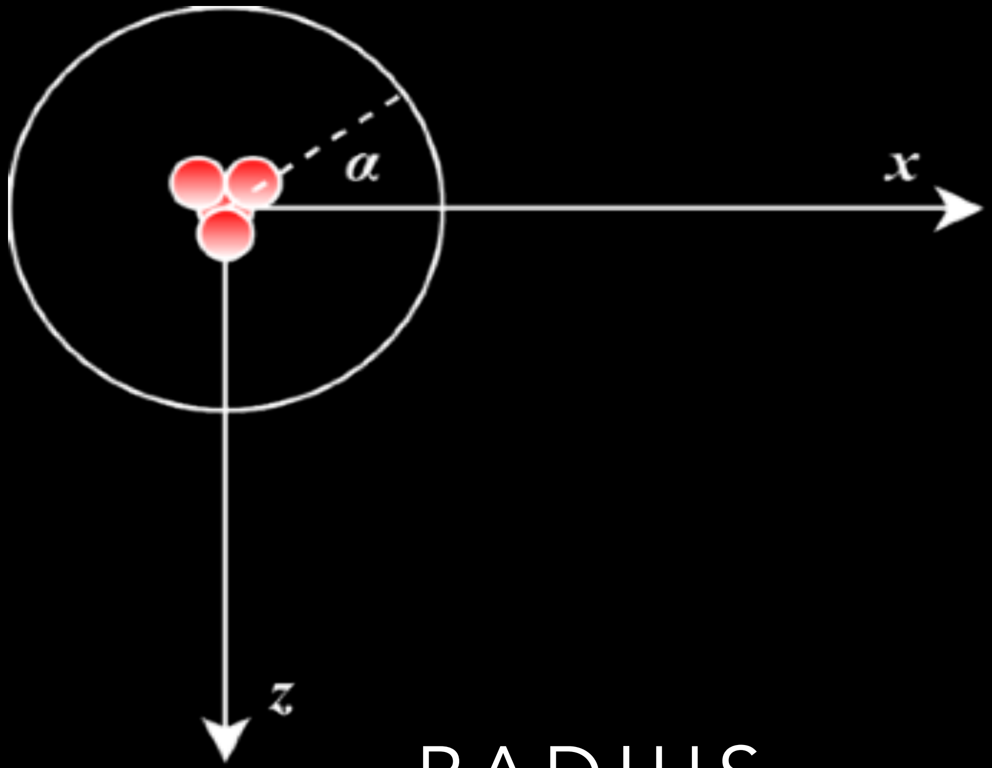
FLAME FORMULATION

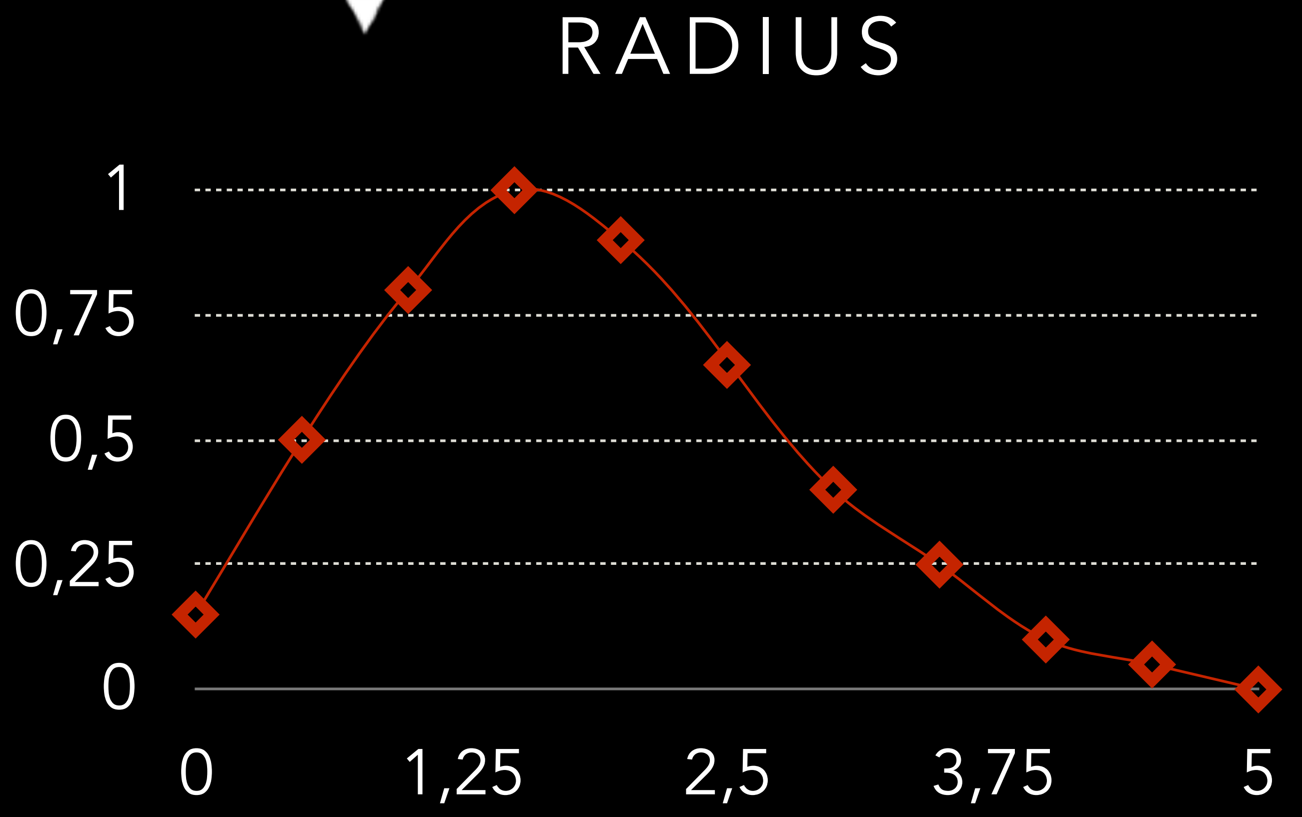
- All N vertices start at same position at *t=0*
- Each one has a random angle α∈*(0,360)*
- Radius follows a curve obtained by regression on a set of hand picked points, depending on the time t. Radius also has a random component used to fill the flame.
- New position at *t=t1* follows equation:

  y = t

  x = cos(α)*r

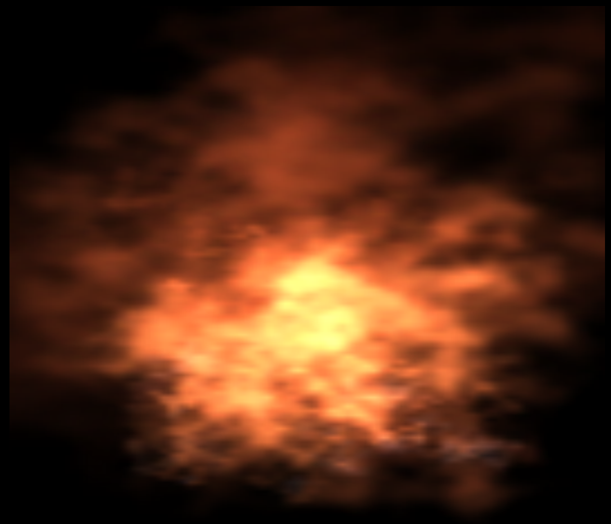
  z = sin(α)*r

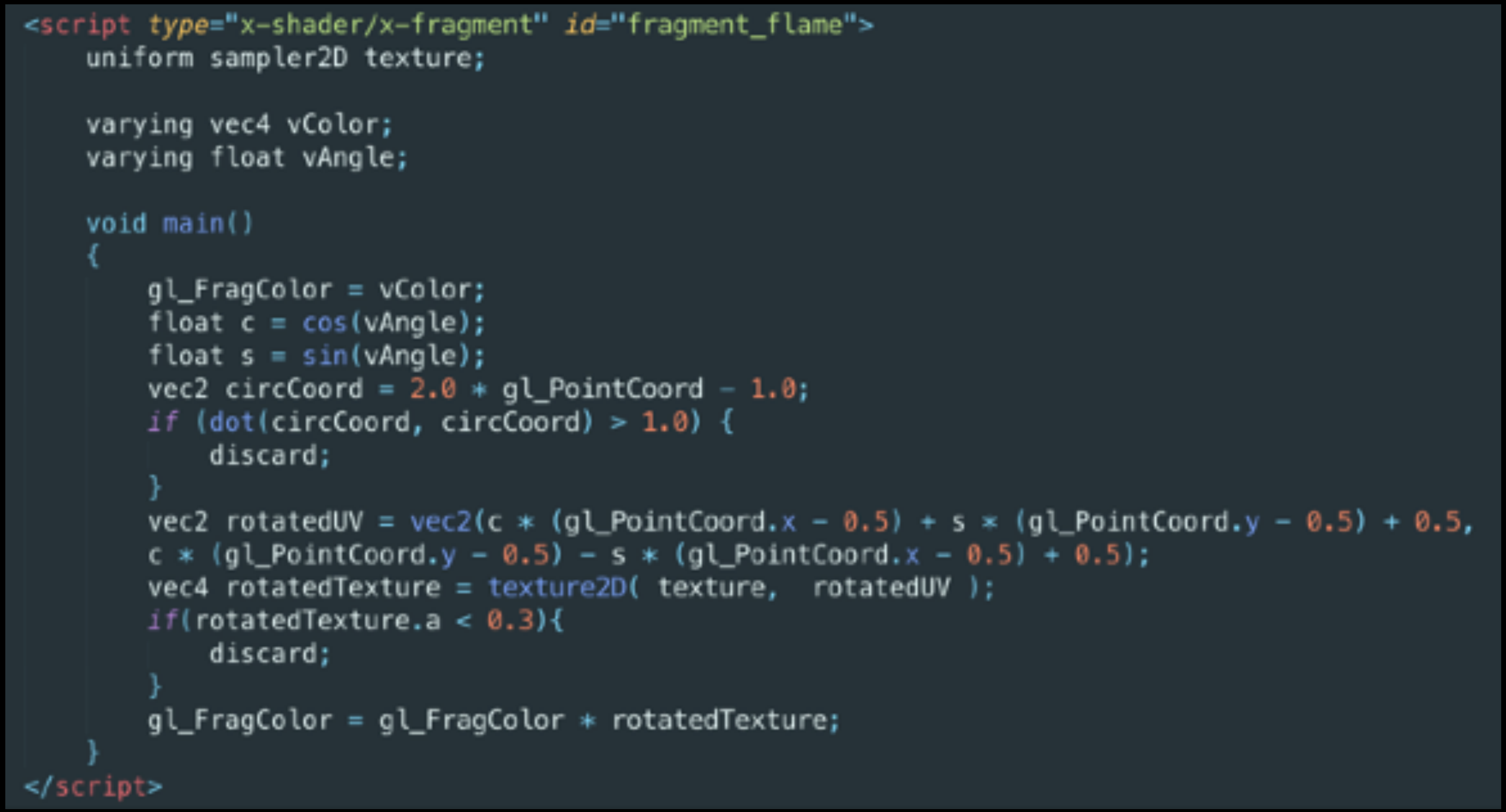




FLAME FRAGMENT SHADER

- Fragment shader is used to manage color and texture shape of flame particles
- A png image with alpha channel is used as texture
- Fragments outside a circle centred in gl_PointCoord are discarded to give an almost spherical shape to the particles
- Texture is centred and rotated according to the particle orientation and coordinates
- Particles are sorted (in the buffer arrays) along the camera view direction in order to make transparencies work





```
<script type="x-shader/x-fragment" id="fragment_flame">
    uniform sampler2D texture;

    varying vec4 vColor;
    varying float vAngle;

    void main()
    {
        gl_FragColor = vColor;
        float c = cos(vAngle);
        float s = sin(vAngle);
        vec2 circCoord = 2.0 * gl_PointCoord - 1.0;
        if (dot(circCoord, circCoord) > 1.0) {
            discard;
        }
        vec2 rotatedUV = vec2(c * (gl_PointCoord.x - 0.5) + s * (gl_PointCoord.y - 0.5) + 0.5,
        c * (gl_PointCoord.y - 0.5) - s * (gl_PointCoord.x - 0.5) + 0.5);
        vec4 rotatedTexture = texture2D( texture,  rotatedUV );
        if(rotatedTexture.a < 0.3){
            discard;
        }
        gl_FragColor = gl_FragColor * rotatedTexture;
    }
</script>
```