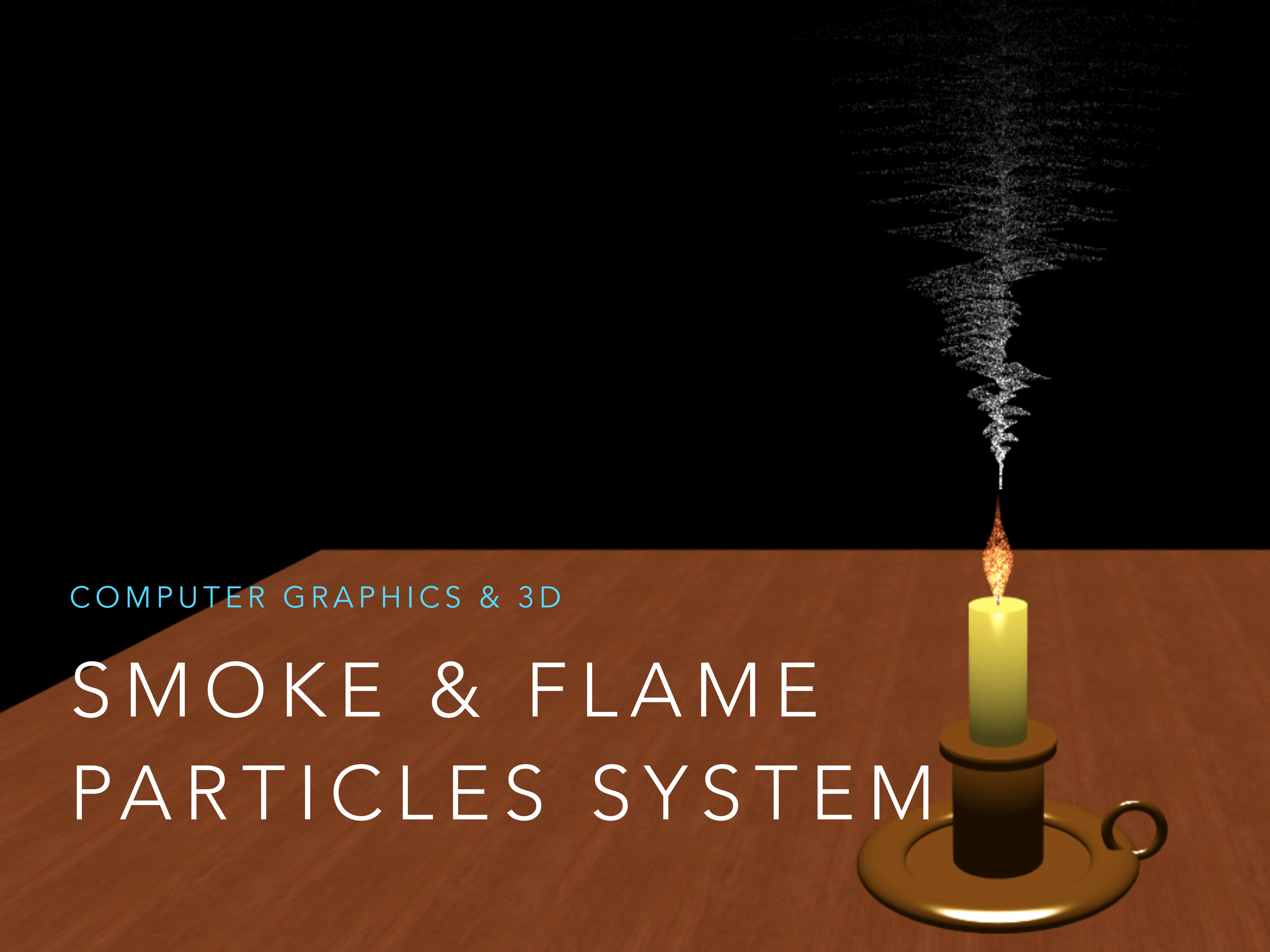


COMPUTER GRAPHICS & 3D

SMOKE & FLAME PARTICLES SYSTEM

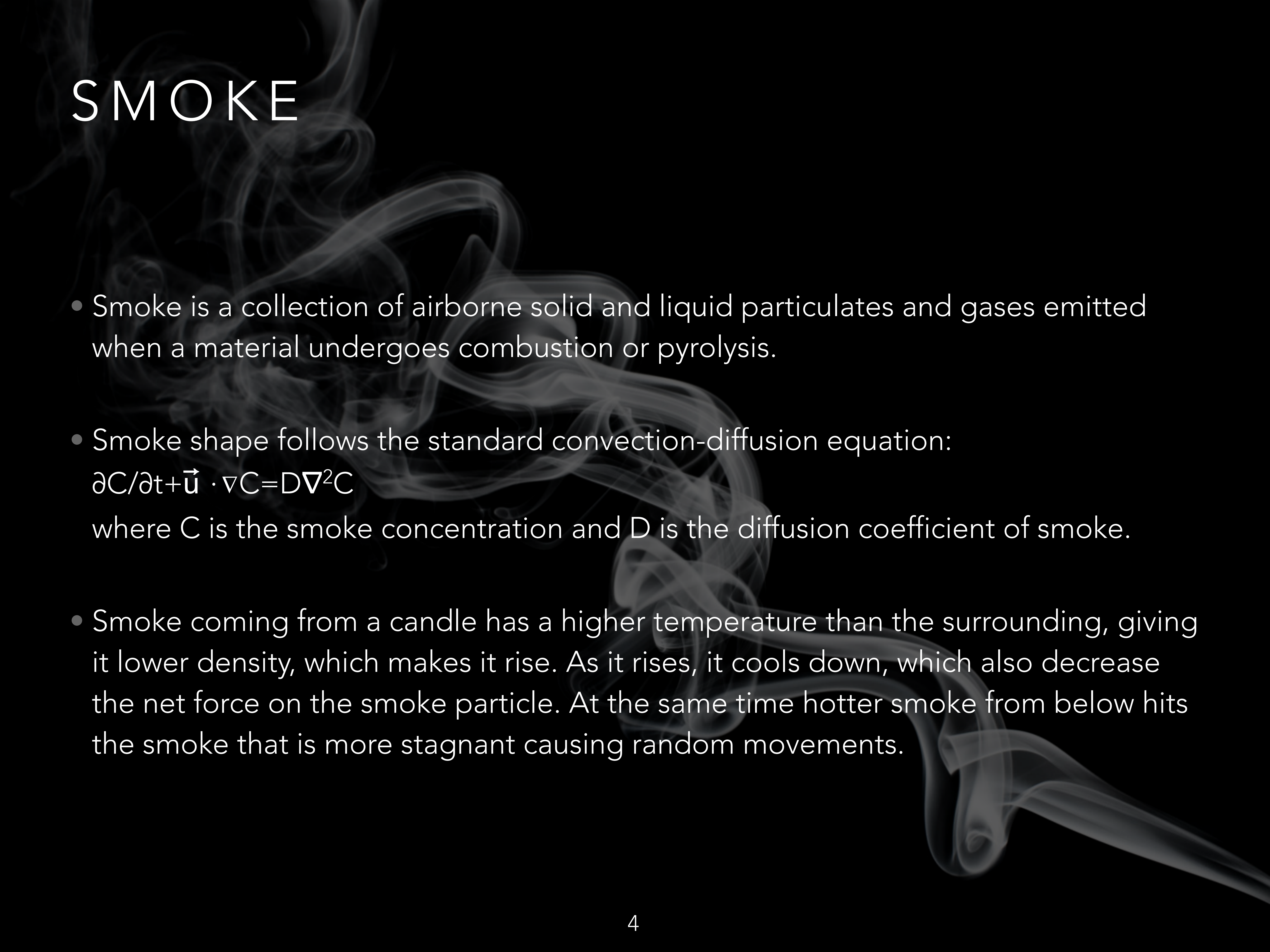
INTENT

- Reproduce the scene of a lit candle
- Flame and smoke represented as particles systems
- Use shaders and technologies studied to model the physical behaviour of the particles

PROJECT INTRODUCTION

- A particle system is a convenient representation of a natural phenomena
- The natural phenomena to be reproduced are Smoke and Flames
- Creation of a pseudo-realistic scene simulating physics using random factors in the particles motion

SMOKE

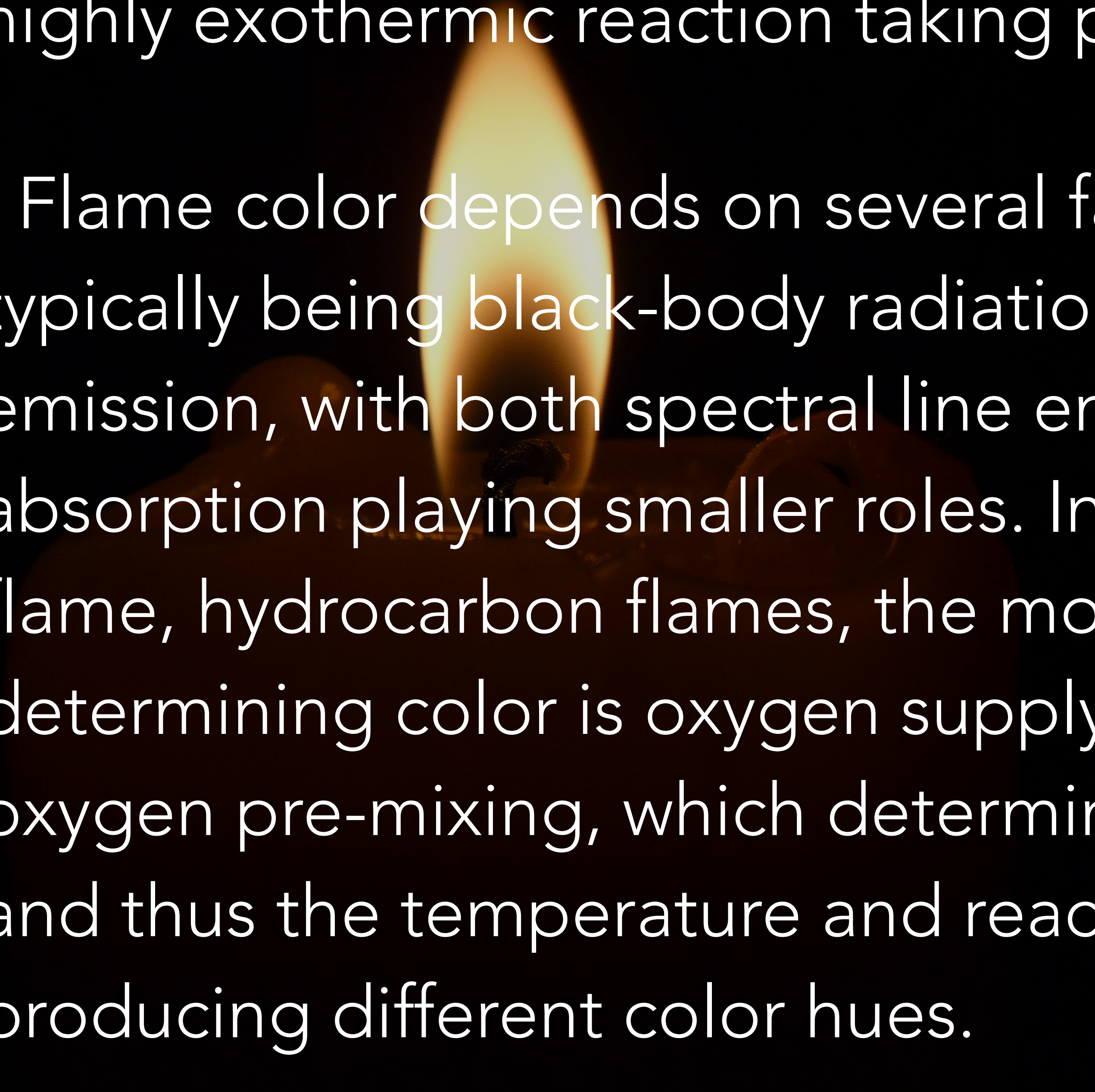


- Smoke is a collection of airborne solid and liquid particulates and gases emitted when a material undergoes combustion or pyrolysis.

- Smoke shape follows the standard convection-diffusion equation:
  $\partial C/\partial t+\vec{u} \cdot \nabla C=D\nabla^2 C$
  where C is the smoke concentration and D is the diffusion coefficient of smoke.

- Smoke coming from a candle has a higher temperature than the surrounding, giving it lower density, which makes it rise. As it rises, it cools down, which also decrease the net force on the smoke particle. At the same time hotter smoke from below hits the smoke that is more stagnant causing random movements.

FLAMES

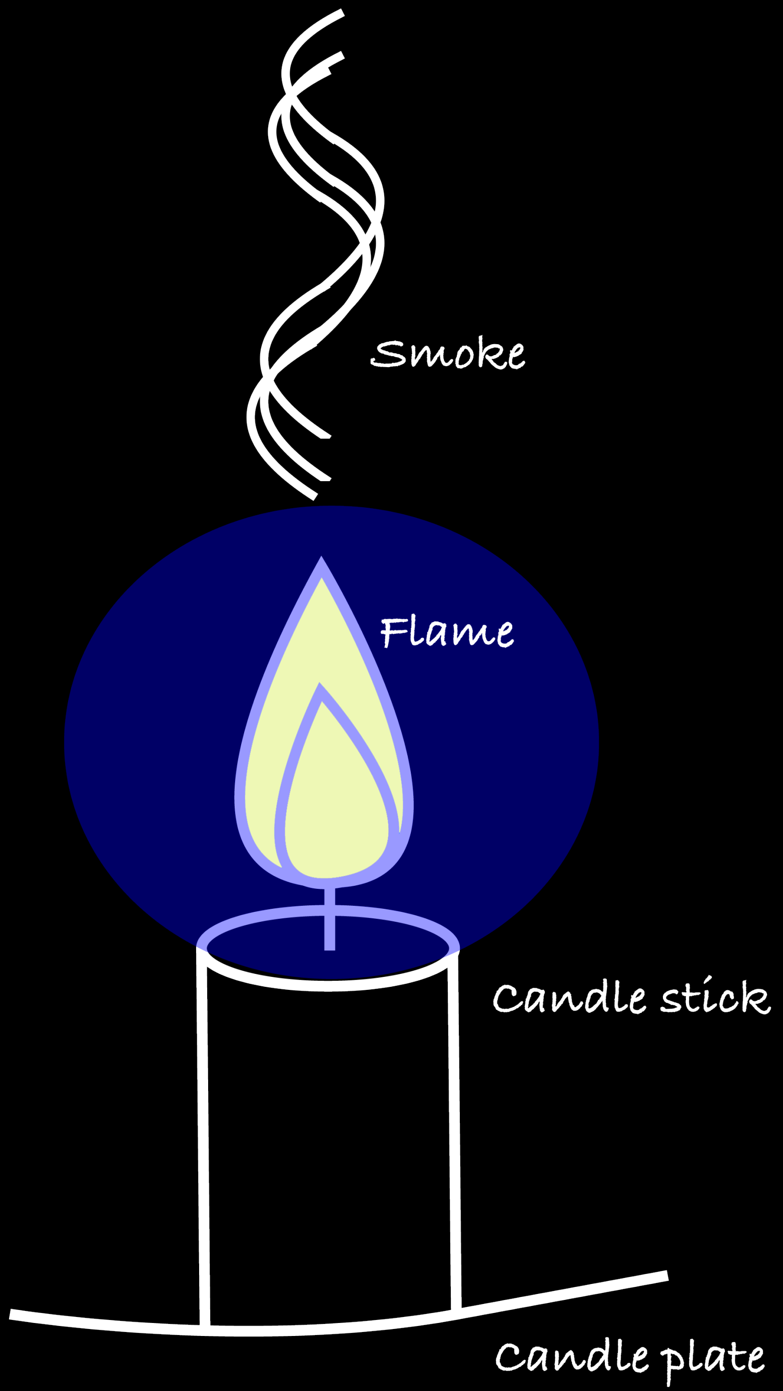


- A flame is the visible, gaseous part of a fire. It is caused by a highly exothermic reaction taking place in a thin zone.

- Flame color depends on several factors, the most important typically being black-body radiation and spectral band emission, with both spectral line emission and spectral line absorption playing smaller roles. In the most common type of flame, hydrocarbon flames, the most important factor determining color is oxygen supply and the extent of fuel-oxygen pre-mixing, which determines the rate of combustion and thus the temperature and reaction paths, thereby producing different color hues.

MOCK UP

- The project will be implemented using WebGL
- *smoke* and *flame* particles will be managed through ***shaders***
  - *The physical model is simplified for this project purposes*
- The scene surrounding *smoke* and *flame* will be represented using ***three.js*** *library*



WHY THREE.js?

- Three.js is an Open Source javascript library that offers methods for interfacing WebGL core
- Three.js allows to create complex 3D animations and system that may be much difficult using only javascript
- A well explained documentation is at: https://threejs.org/docs/

HOW THREE.JS WORKS

- An object is defined by a Geometry and a Material. Both class are available in lots of specialisations.
    - Geometry is a set of vertices, disposed to represent a certain object.
    - Material defines object's properties (brightness, shadowing, texture, etc.).
- Moreover other libraries are available to help managing camera or movements (Orbit Controls, Tween, etc…).

All seems easy and amazing, but:

- Geometry and Material classes carry on a useless baggage of informations.
- Fortunately it's possible to define our own geometry and material using custom vertices sets and shaders.
- This strongly increase performances in a simple context like this (if all done correctly).

PROCEDURE

Shaders

- Vertex Shaders
  - Attributes
    - particle starting position
    - particle size
    - particle trajectory angle
    - time offset (for continuous generation)
  - Uniforms
    - time t
    - time life (before regeneration)
    - speed
    - opacity
- Fragment Shaders
  - Color
  - Texture

Three.js

- initialize scene
  - camera
  - light
  - box containing the scene
  - table
    - geometry
    - texturing
  - load candle obj created with blender
  - init of smoke and flame Geometry
    - setting first position e angle
    - linking attributes with shaders
    - setting and linking uniform variables
  - Audio

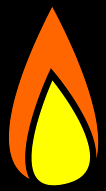
SCENE

Table

- *PlaneGeometry*
  - static as floor
- *MeshBasicMaterial*
  - *table texture style*

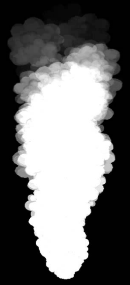
Candle

- Hand crafted in blender
- imported with LoadingManager
  - positioned on the table



Flame

- BufferGeometry
  - defined as a set of N vertices with *attributes*:
    - position
    - customSize
    - customAngle
    - timeOffset
- ShaderMaterial
  - move positions per flame-like shaping
  - texturing
  - color gradient



Smoke

- BufferGeometry
  - defined as a set of N vertices with *attributes*:
    - position
    - customSize
    - customAngle
    - timeOffset
- ShaderMaterial
  - move positions per smoke-like shaping
  - texturing
  - color gradient

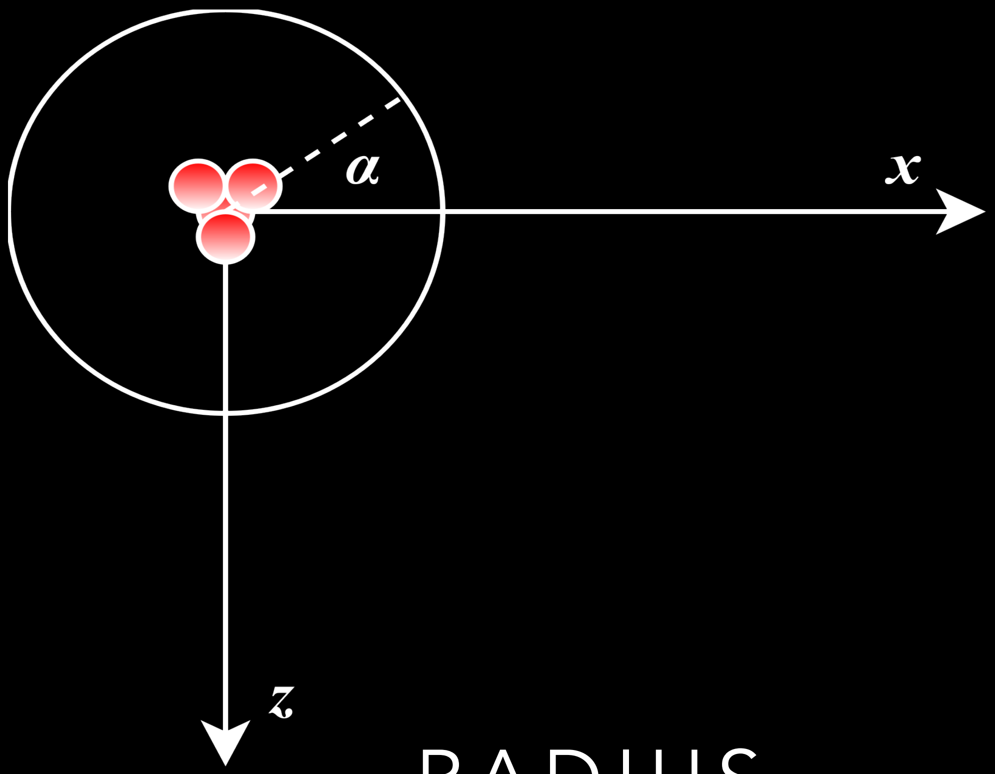
FLAME FORMULATION

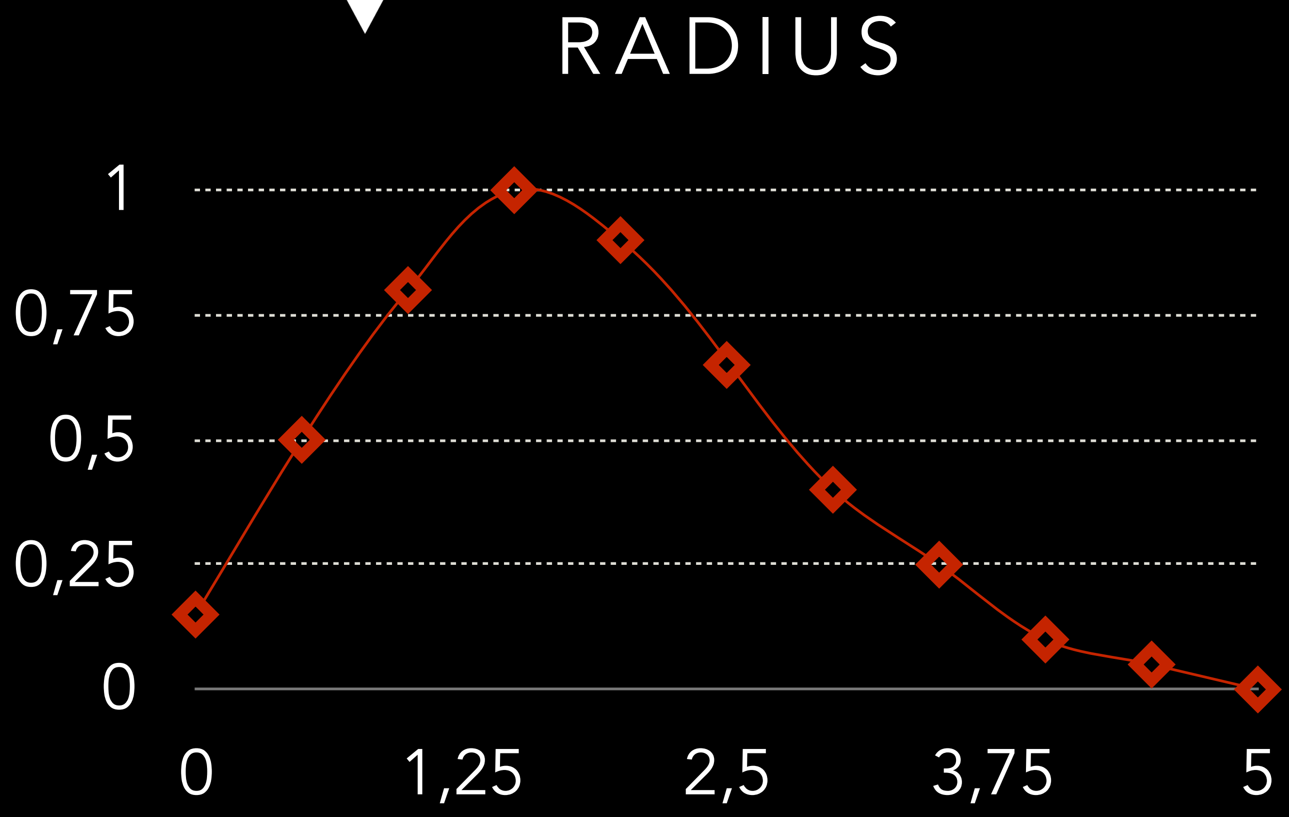
- All N vertices start at same position at *t=0*
- Each one has a random angle α∈*(0,360)*
- Radius follows a curve obtained by regression on a set of hand picked points, depending on the time t. Radius also has a random component used to fill the flame.
- New position at *t=t1* follows equation:

  y = t

  x = cos(α)*r

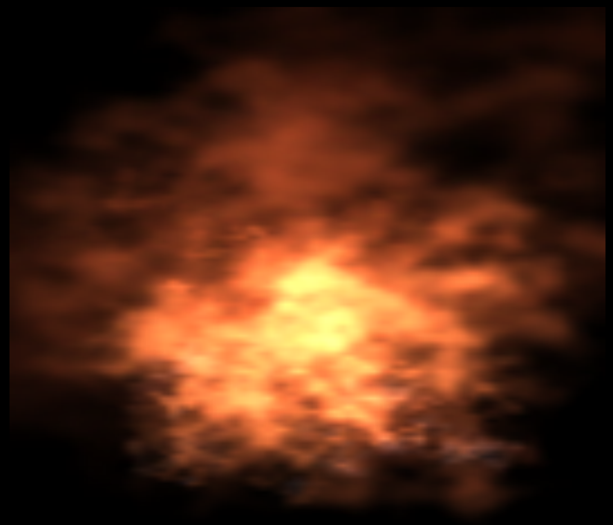
  z = sin(α)*r

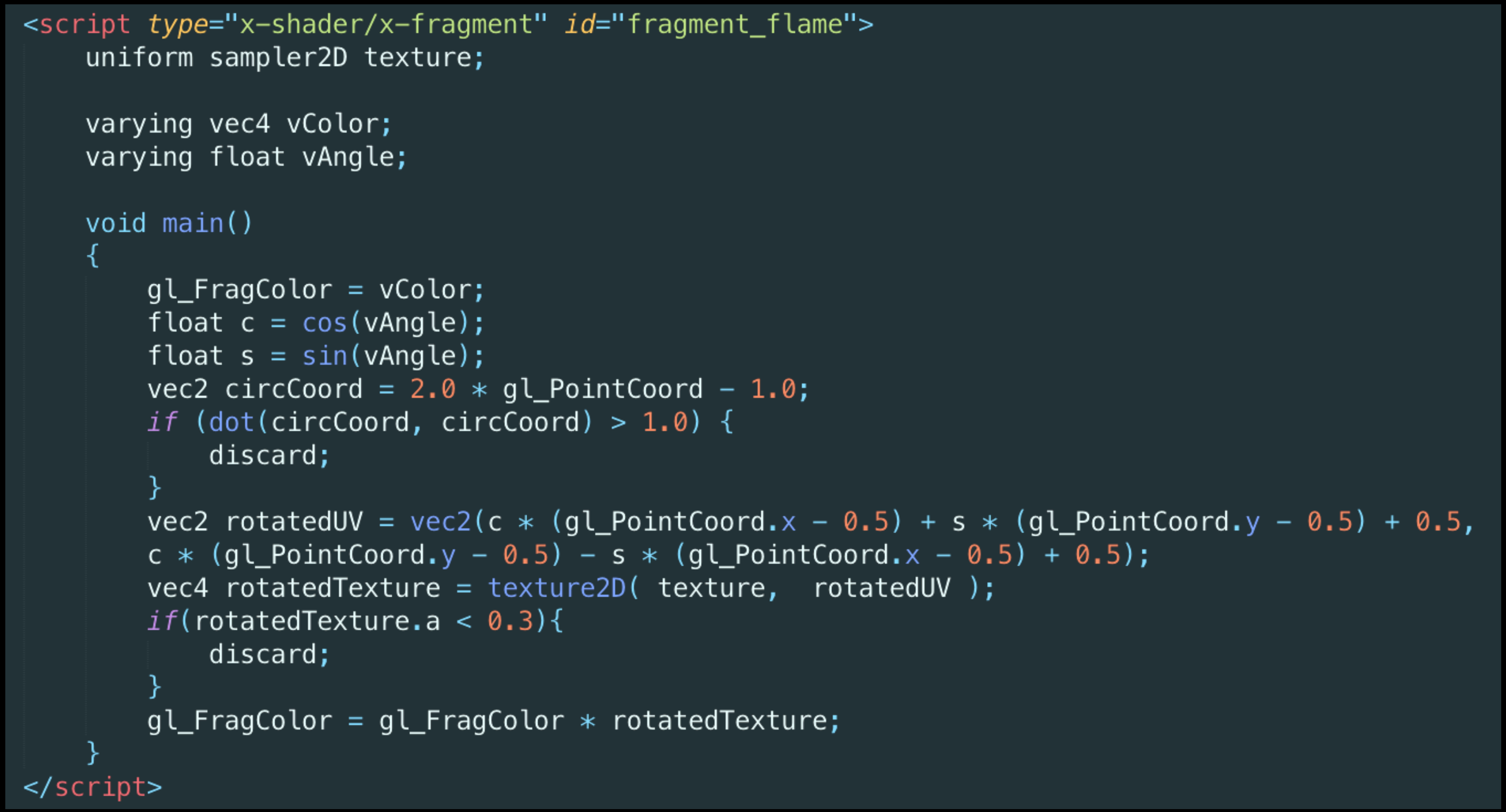




FLAME FRAGMENT SHADER

- 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

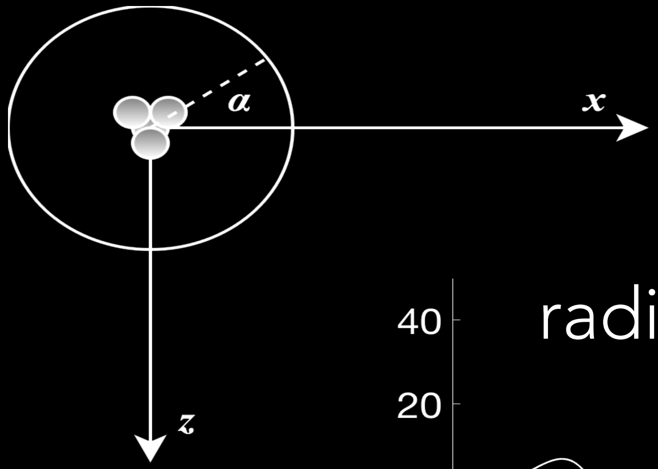


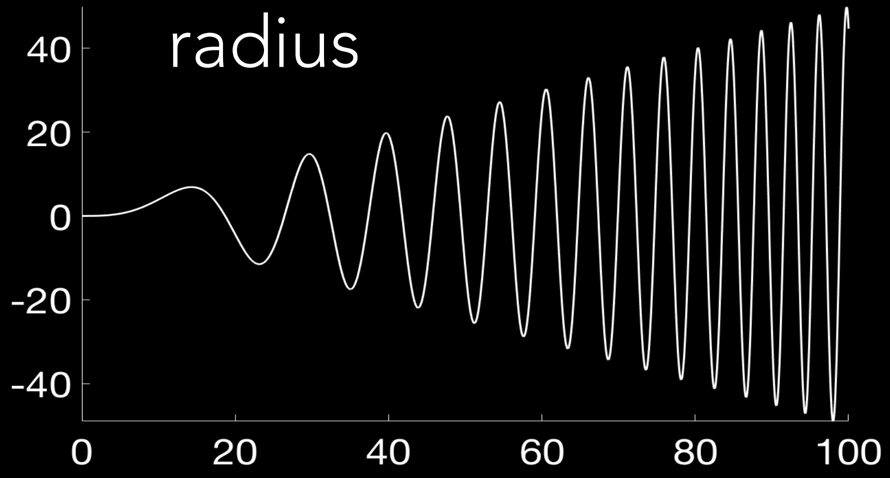


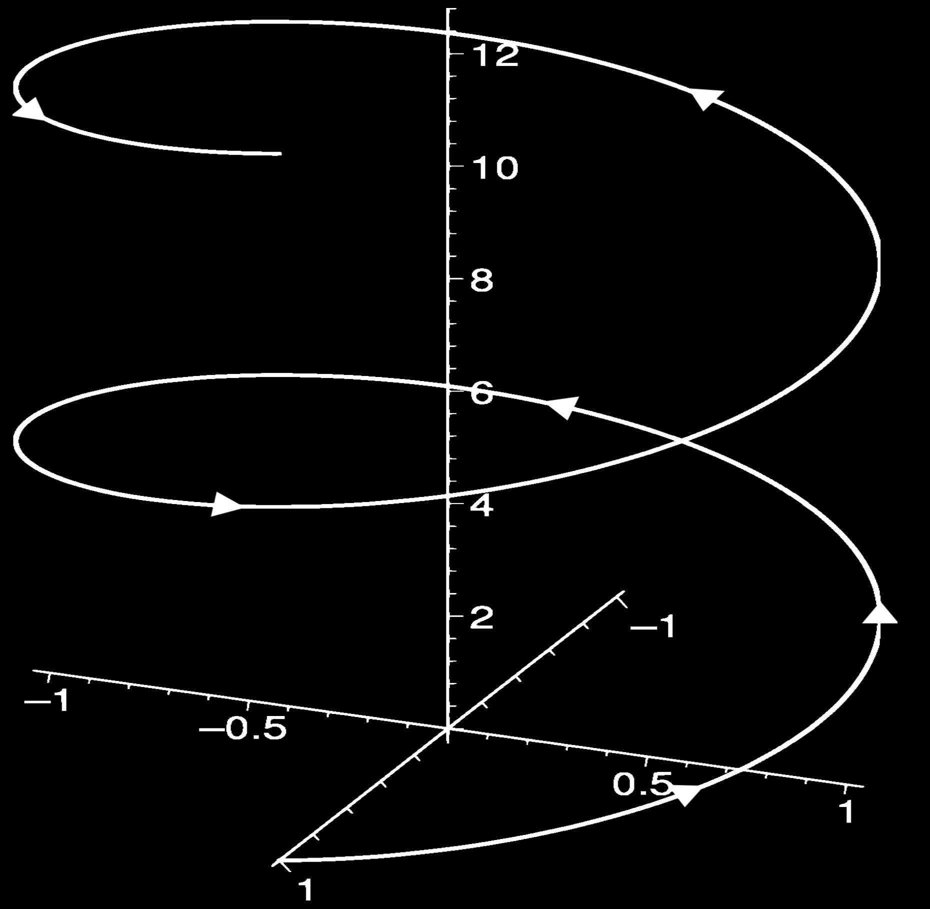
```
<script type="x-shader/x-fragment" id="fragment_flame">
    uniform sampler2D texture;

    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>
```

SMOKE FORMULATION

- Similar to flame model
- Radius:
  $0.5*(x)*\sin(0.009*x^2)$
- New position at *t*=*n* follows equation:
  $y = t$
  $x = r*\cos(t)+t*\cos(t)^2+\sin(t)$
  $z = t*\sin(t)+t*\sin(t)^2$
- High random component is added

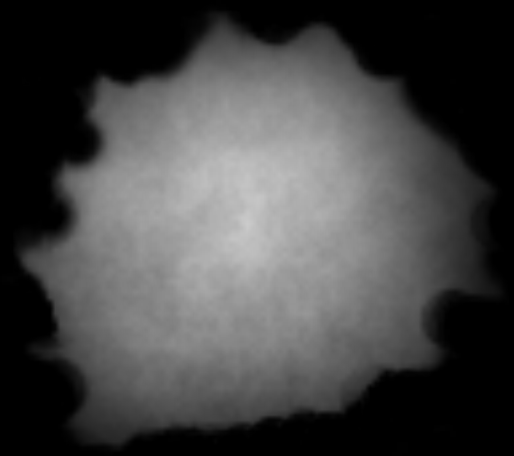


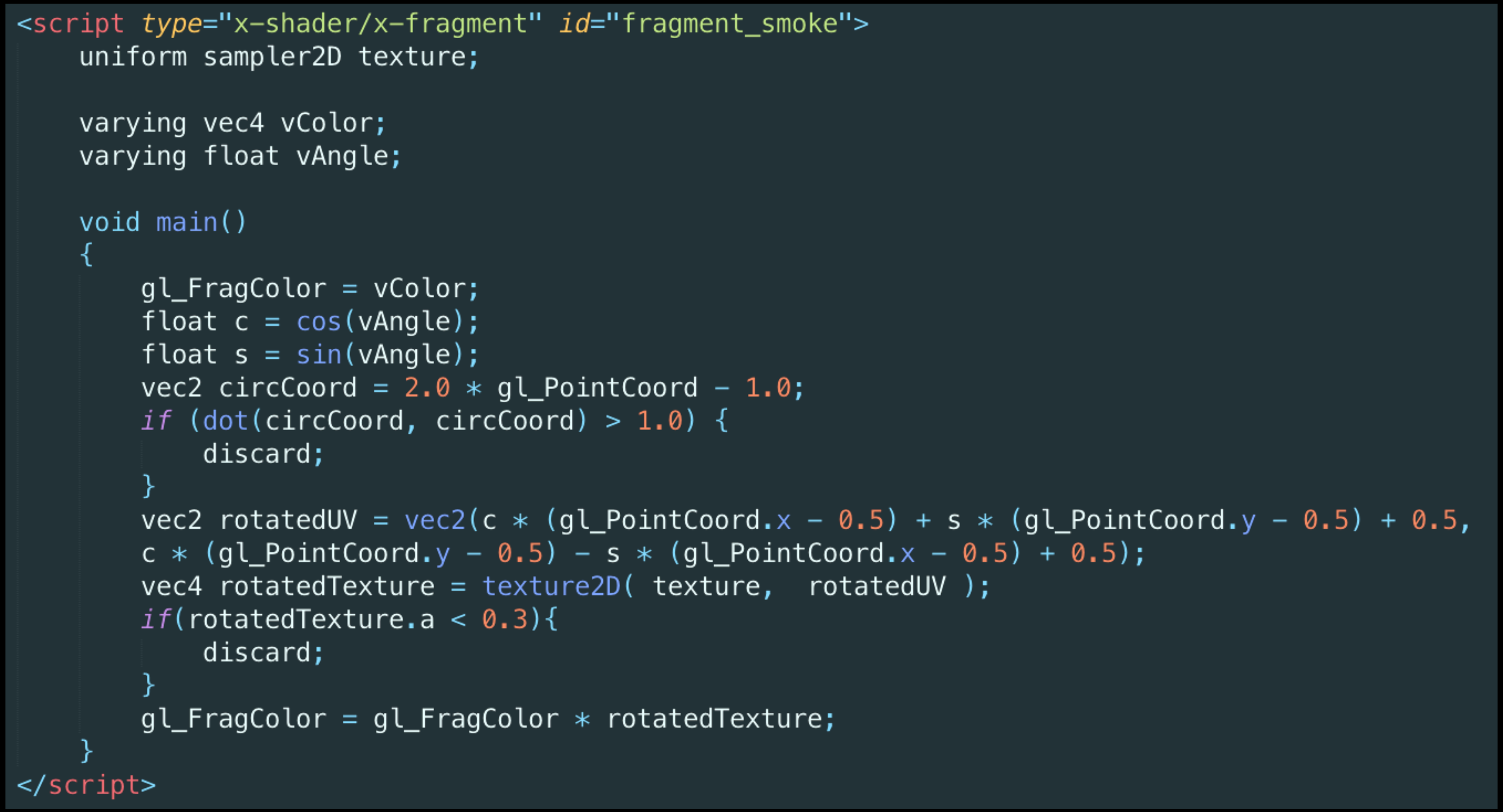




SMOKE FRAGMENT SHADER

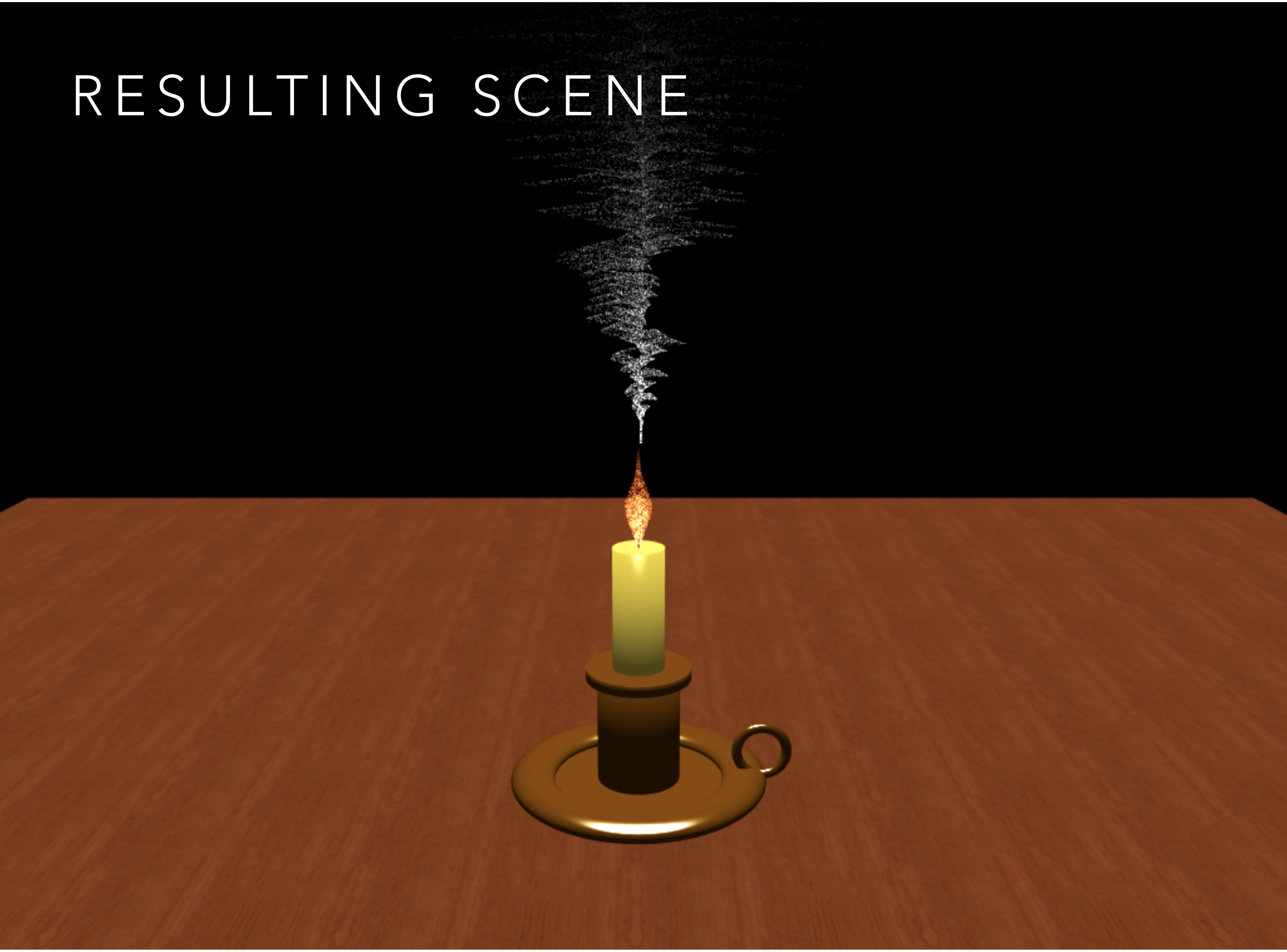
- Fragment shader is used to manage color and texture shape of flame particles
- A png image with alpha channel is used as texture
- Fragments outside a circle centred in gl_PointCoord are discarded to give an almost spherical shape to the particles
- Texture is centred and rotated according to the particle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





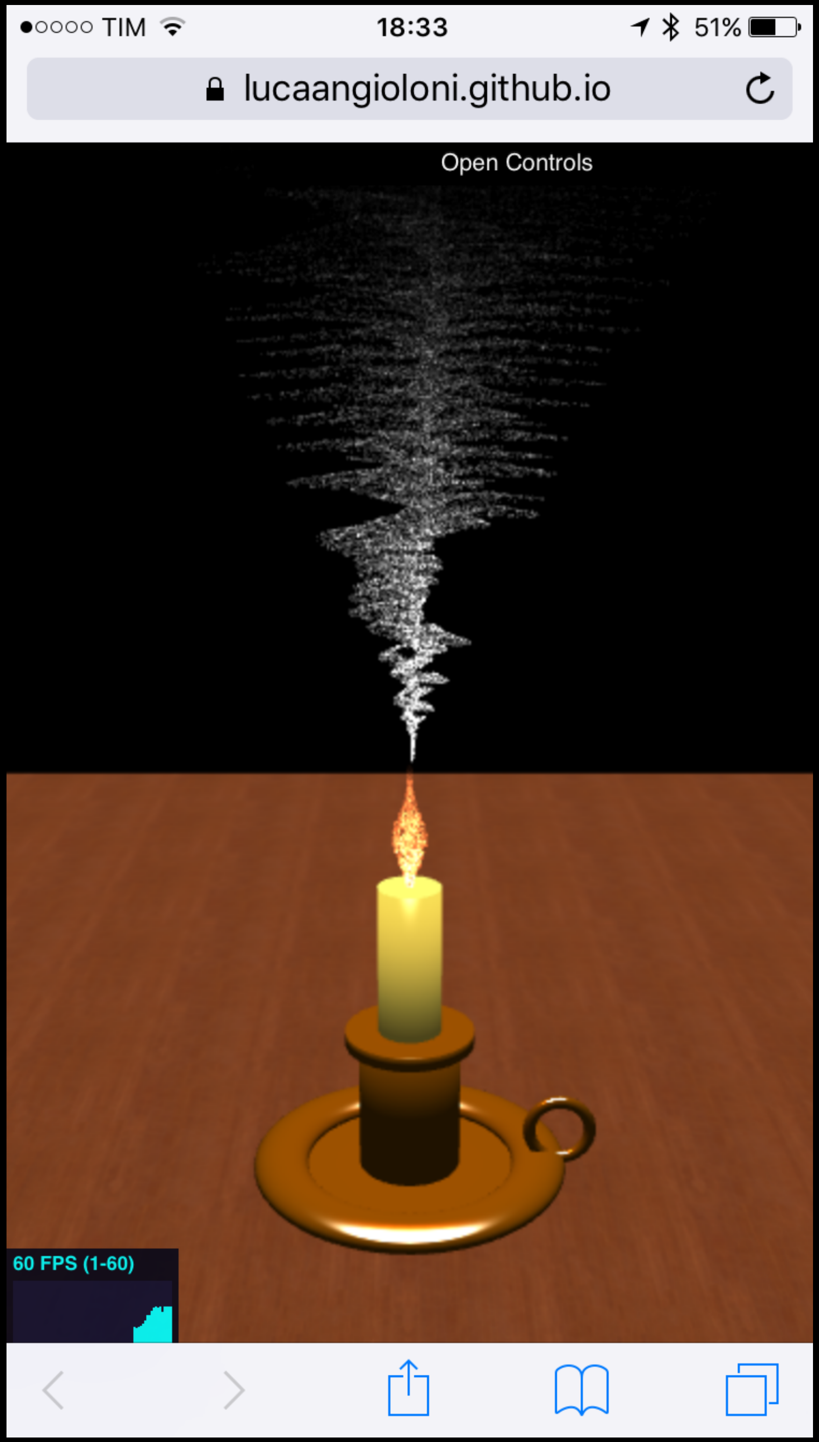
```
<script type="x-shader/x-fragment" id="fragment_smoke">
    uniform sampler2D texture;

    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>
```

RESULTING SCENE



MOBILE PORTABILITY

- New mobile device are optimised for graphics operations. This allows desktop-like performances.
- The project has been made mobile ready with touch event controls and great performances.
- Even on older generation devices ~60 fps are rendered.



PERFORMANCES

- Using a MacBook Pro and Safari as reference:

Smoke Vertices	Flame Vertices	Fps
50K	15K	60
100K	15K	60
100K	40K	60

- Using an iPhone 6S and Safari as reference:

Smoke Vertices	Flame Vertices	Fps
50K	15K	60
100K	15K	60
100K	40K	60

CONCLUSIONS

- An example is reachable at : https://lucaangioloni.github.io/SmokeGL/
- The system can reproduce different scenario, the only difference would be in formulas describing the event