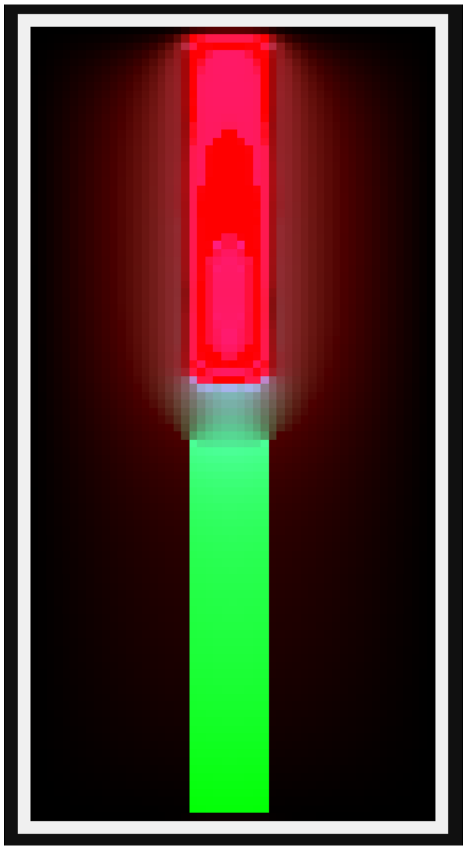
Candle combustion simulation
Warsaw University of Technology

Damian Legutko, Katarzyna Wiater

June 2023



Contents

1 Introduction

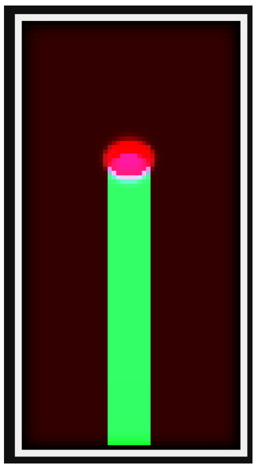
Goal of the project was to create simple real-time candle combustion simulation using FEM (cellular automata). Simulation is hosted from Flask and can be viewed using web browser (http://localhost:5000).

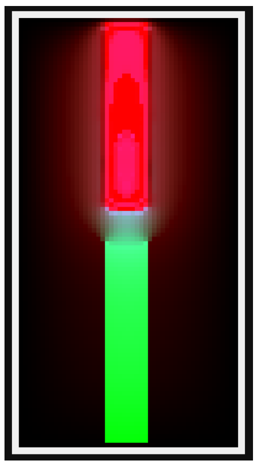
2 Boundary conditions

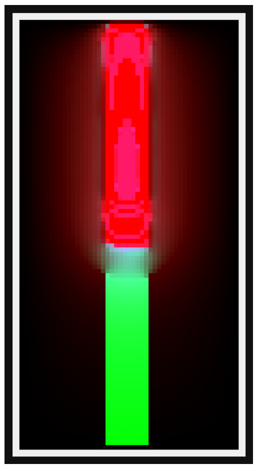
Ignition is started by high temperature region at the top of the candle (only in the first frame of simulation).
Cells at the edge of canvas have their properties set to 0 before every simulation step.

3 Simulation display

Each pixel of the display is colored with following rules:
- the higher the temperature - more red color
- the more material inside the cell - more green color
- the more material combined with high temperature - more blue color


Ignition

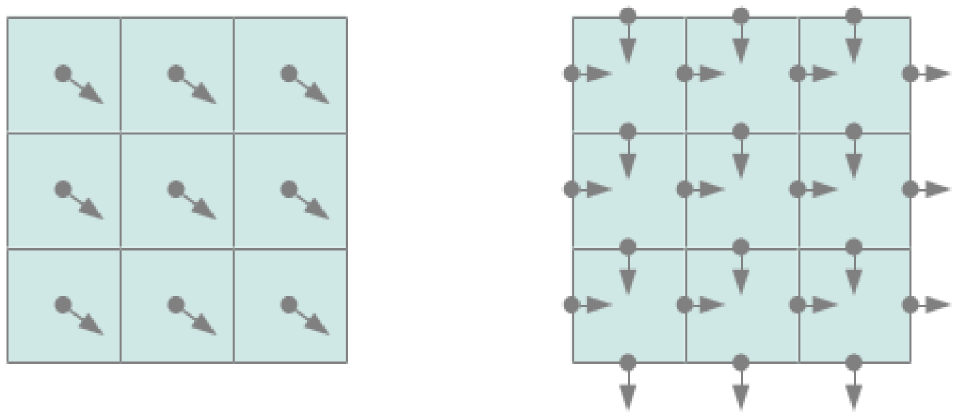

Burn at $t \approx 5s$


Burn at $t \approx 8$

4 Grid definition

Simulation is based on a staggered grid of egdes and cells, where each cell has four edges (neighbor cells share edges). Initially we planned to implement proper fluid simulation but solving divergence equations was too big of a challenge, thus

currenly, edges only store velocity information.
Each cell contains following properties:
- *d* - amount of material inside the cell
- *dd* - rate of change of material amount
- *T* - temperature of the cell
- *dT* - rate of change of the temperature
Rate of change properties are only needed when calculating next simulation step and they are not displayed.



Staggered grid - Left: cell positions, Right: edge positions

5 Equations

Simulation uses few equations to solve grid states at each time-step. Each equation contains fine-tuned constants. Unfortunately due to high CPU loading, simulation is highly likely to be unstable if one of the constants is changed or the time-step changes.

5.1 Turbulence

Turbulence is implemented as a noise on velocity values when material reaches specific temperature.

```
def turbulence(cell):
    if cell.T >= 60:
        cell.edge(Edge.BOTTOM).v += (random()-0.5) * 0.05 * dt
        cell.edge(Edge.LEFT).v += (random()-0.5) * 0.1 * dt
    return cell
```

5.2 Gravity / buoyancy

Once material temperature reaches specific value, material experiences upward velocity proportional to its temperature.

```
def gravity(cell):
    if cell.T > 60:
        cell.edge(Edge.BOTTOM).v -= dt * 0.2 * cell.T
    return cell
```

5.3 Combustion

Once material reaches combustion temperature, its temperature grows even more but material amount goes down.

```
def combustion(cell):
    # combustion and evaporation
    if cell.T >= 100 and cell.d > 0:
        cell.dT += 4000.0 * cell.d * dt
        cell.dd -= 16000.0 * cell.dT * dt
    return cell
```

5.4 Diffusion

Diffusion is implemented a difference between average value of cell neighbors. It's used to spread heat, and the material once it reaches vapor temperature.

```
def diffusion(cell):
    # vapor
    if cell.T >= 30:
        cell.dd += 0.96 * cell.diffuse(dt, lambda _cell: _cell.d)
    # heat diffusion
    cell.dT += 1.4 * cell.diffuse(dt, lambda _cell: _cell.T)
    return cell
```

5.5 Advection

Advection is a process of moving cell properties in the direction of velocity. In simulations it plays an important role of moving heat and material upwards.

```
def advection(cell):
    cell.dd += cell.advect(dt, lambda _cell: 0.1 * _cell.d)
    cell.dT += cell.advect(dt, lambda _cell: 0.1 * _cell.T)
    return cell
```

6 Simulation behaviour

Using those simple rules above, many real-life behaviours can be seen in the simulation, for example:
- if combustion temperature is too low or burn rate too high, the flame dies out
- if the gravity is too high, the flame stretches and is more likley to die out
- flame is composed of few sections, with varying temperature
- flame flickers like a real one

Obviously simulation is far from perfect, but it shows what can be achieved with only few lines of code.

7 Future development

Initially we planned on implementing Cantera into the simulation as a base for solving combustion. However that would require implementing more materials to the simulation and solving Cantera will be much more CPU intensive. Example implementation of combustion would look something like this:

```
gas = ct.Solution('gri30.xml')
reactor = ct.Reactor(gas)
reactor_network = ct.ReactorNet([reactor])
...

def combustion(cell):
    gas.TPX = cell.T, cell.P, cell.mixture_str
    end_time = reactor_network.time + dt
    while reactor_network.time < end_time:
        reactor_network.step(end_time)
    # Extract updated state
    cell.T = reactor.thermo.T
    cell.P = reactor.thermo.P
    cell.mixture_str = ", ".join(f"{s}:{x}" for s, x
                             in zip(gas.species_names, reactor.thermo.X))
    return cell
```

8 Literature

- Shahriar Shahrabi
Gentle Introduction to Realtime Fluid Simulation for Programmers and Technical Artists
https://shahriyarshahrabi.medium.com/gentle-introduction-to-fluid-simulation-for-programmers-and-technical-artists-7c0045c40bac

- Gonkee
But How DO Fluid Simulations Work?
https://youtu.be/qsYE1wMEMPA

- The Coding Train
Coding Challenge 132: Fluid Simulation
https://youtu.be/alhpH6ECFvQ

- Ten Minute Physics
17 - How to write an Eulerian fluid simulator with 200 lines of code.
https://youtu.be/iKAVRgIrUOU