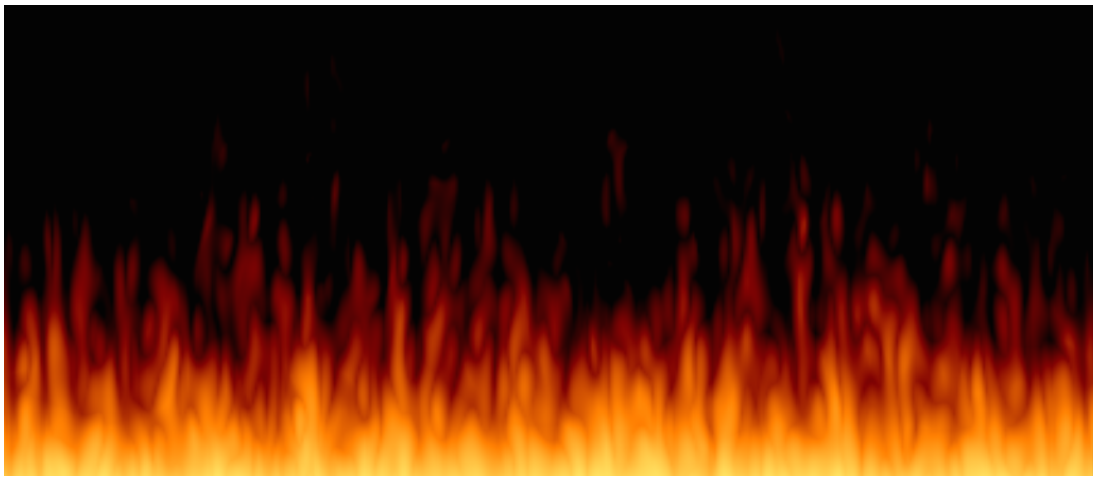
# "I see fire"

Procedural methods for images, TNM084

Linköping University

---

Linnéa Mellblom linme882



January 16, 2015

## 1 Introduction

Turbulent phenomenas is interesting and often pleasant to look at. By this thought the project took form; to create procedural fire.

## 2 Goal

The goal with the project is to animate ”realistic” (as much as it can be) fire in real time. Main goal is to make it look nice in 2D and if time is enough, develop and go to 3D instead. The priority is to make the fire look good, rather then just take it to 3D quiet fast.

The aim is to use different kind of noise functions to create procedural fire. The implementation is to be written in WebGL and shaders in GLSL. The language is chosen to make me learn more and discover something new, but also to be able to show the result to a wider range of people.

## 3 Noise functions

Noise can be generated and manipulated in many different ways to create visual effects. It is used to increase the appearance of realism in computer graphics, since objects do not have a perfect surface or adding randomness of the structure of a cloud or a fire.

Classics noise such as Perlin noise, is a gradient noise which means that a pseudo random gradient is associated with points that are regularly spaced. The noise function mainly consist of interpolation of the closest grid point and their gradients. When picking the gradients, they need to have enough variation to conceal the fact that the function is not truly random. The gradients need to be pseudo random for the noise function to be repeatable, i.e always yield the same value for a given input point [2].

To overcome some problems with Perlin noise, Simplex noise was developed with a lower computational complexity. Most important change are the simplex grids. The Perlin noise algorithm has a regular square grid in 2D while simplex grid uses the simplest and most compact shape that can fill the space, and therefore in 2D a triangle. This reduces the amount of computations since Simplex noise uses this strategy. Another improvement involves the interpolations in Classic noise along each dimension. As the dimension increase, the more complex computations since the computation of the analytic derivative of the interpolated function. In Simplex noise a straight summation of contributions is instead used.

Flow noise is a flow texture that is based on Perlin- or Simplex noise. Flow noise boils down to two ideas: rotate the underlying gradients at each grid point in the noise function, to get a swirling, flowing look and push smaller features away from center of large features (as real turbulent flow, a small swirl appear on a edge of a larger swirl and therefore give a more realistic look). The trick is to evaluate the sum of different scales and displace the domain for the next smaller scale along the gradient at the evaluated noise function so far. [3]

## 4 Implementation

This project is written in WebGL and GLSL. This is due to that is easy to display the result to everyone and an opportunity for me to learn WebGL a bit more.

In the project I worked with both *flow noise* and *simplex noise* (short described above). Mainly to see if there are differences and to be able to experience more of noise.

The most is written in the fragment shader. Here I calculate each color that the pixel will have. The WebGL setup took a while to implement but with great tutorials the job got easier [1]. The hard part is to look at the code and understand what it does at each step. In the vertex shader I only transformed the origin to the left bottom corner and also mad it possible to add shapes in pixel width rather then just -1 to 1 as standard.

To be able to get the noise to look like fire, I added several octaves. Since the program shows 3 different fires, everyone has different kind of combinations. The main thing that is common, is that the frequency is higher in x than y. This due to that a flame is often higher than wider (or at least that was what I was aiming at). By taking a higher frequency in x, the denser the pattern will be in x and therefore it will look like the pattern is higher in y.

Here below are one example of from one of the function that generates the fire pattern. The 0.5 defined the amplitude of the noise. *p.x* is the position in x, *p.y* the position in y and *t* is the time. The *snoise* is using the simplex noise. The frequency in this case is 16 in x and 4 in y. That will make the pattern more stretched out in y. Also the time is added to the y component, which will make the flames move up. The last argument is *0.3*t*, this argument will make the pattern update (change) in time.

```
n = 0.5 * (snoise(vec3(16.0*
    p.x,4.0*p.y+0.6*t,0.3*t)));
```

One crucial point to take in consideration, is that the when adding different octaves, the different noise need to be different amplitudes and that the animation in y is different. This will make the flame move in different kind of motion and look more like a turbulent phenomena.

To make a lite more interesting result, I decided to make some circles (or ellipses) cut out from the texture depending if the user wants to. In order to do so,I created a check pattern (how many depending on the user input) and then calculated a circle ($tileCoord.x * tileCoord.x + tileCoord.y * tileCoord.y$) in each of these and then added that to the fire color.

One important aspect of fire, is when it rise it should be more perturbed. The fire will fade away more when it gets higher. Therefore the perturbation depends on the y-value of the texture that we are rendering on. The higher we are, the more perturbation we add.

One thing I have just but a little time is that it will look okay on mobile phones also. So that the canvas is depending on the size of the browser.

## 5 Result and benchmarks

Here are some benchmarks for the project and the evolving of the fire. It took quiet some time between these iterations to get to the final result.

Here I have only showed the iteration of the fire that used the simplex noise. They have also different kind of combination of octaves.

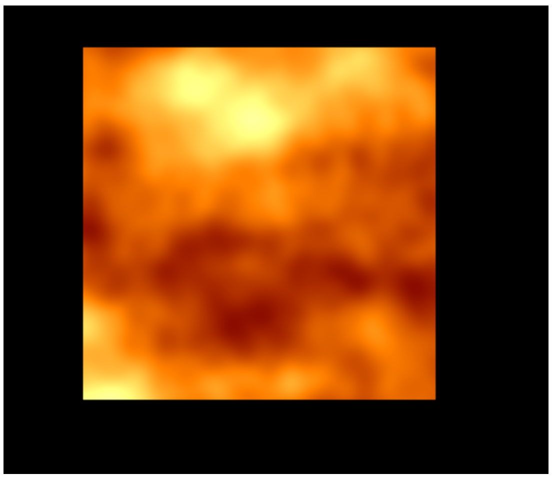


Figure 1: *Early version.*

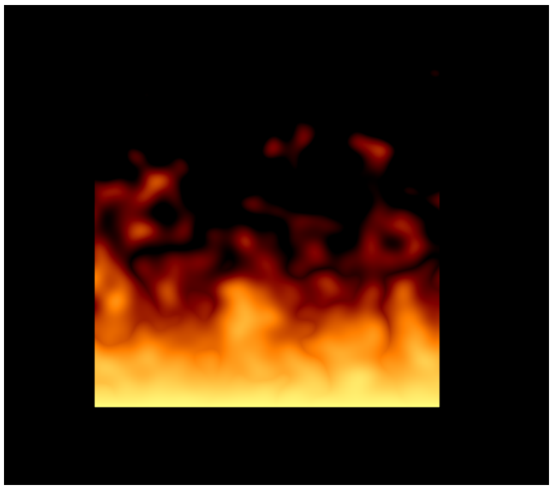


Figure 2: *Tried different kind of shapes.*

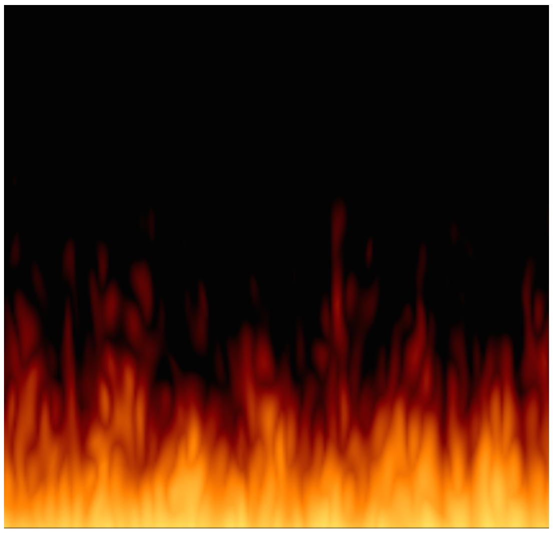


Figure 3: *When it started to look more like flames.*

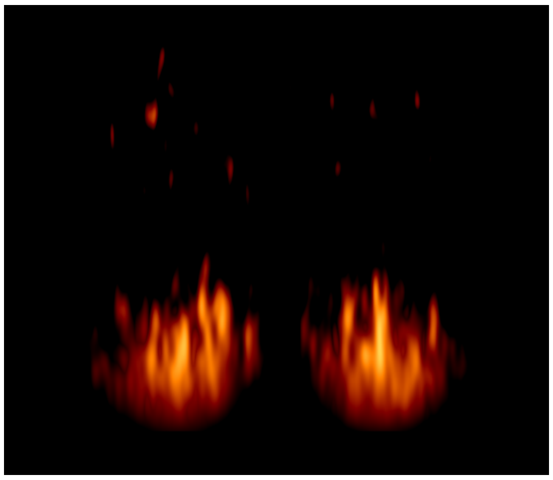


Figure 4: *A snapshot of the final result.*

## 6 Discussion

The use of procedural noise textures for simulation of a turbulent phenomena provides a highly customizable animation framework. The appearance of the fire are determined by only a few parameters like frequency, amplitude and how much the fire should flicker. If time is spent on these variables and tweaking, the texture can resemble fire quiet well. But, by for example using the flow noise, it is possible to define flows that do not resemble any true fluid at all if the variables is combined in some ways.

It took quiet some time to tweak these variables and I could easily continued much further. It is hard to say that now have I reached to a point where I am fully satisfied. In the beginning I thought that it should not take that long to actually find the "right" variables and combinations, but further in the project i realized that I needed to focus on just the 2D fire and leave the 3D to a project later on instead. Since the result should resemble fire quiet well, I chose to take the time on the variables instead. So sometimes it will not look like much code, but instead it is very much time spent on these variables and combination. I have spent several hours on adding different frequencies, mixed different noises and so on.

The color of the fire was hard to get right and I think i have not *nailed* it yet. These are also some variables that can be tweaked until you get a near perfect result. Eventually I stuck with the variables that I have and said that this was fairly good.

By using both Simplex noise and Flow noise, I must say that I think that Simplex noise was easier to get a good result with. Flow is nice, but a bit harder to actually look good. By only accidentally changing one variable, the result could be unexpected and not resemble any true fluid at all. You had to be more careful when assigning variables and adding different frequencies together.

I am almost happy with my result. I can put several hours more on this and it is hard to know when to stop and say that you are finished. Maybe I put to much time on the tweaking of variables, instead of getting it to 3D. But that is hard to say if it had been a better result from there.

The best result in my opinion, is when you use Simplex 1, cut out circle of three in width and two in length. Sadly, some combination will make the fire look more terribly. But that only shows that with only a few variables changing, the whole result can look different.

Since I not have worked so much in WebGL before that was a new challenge. With some tutorials I could pick out the important stuff and modify so that it would fit my purpose; things like how to send variables to the fragment shader, how to add object and so on.

From this project I have learned a lot. Not only WebGL but also how to think when starting from scratch. The most fun thing with these kind of programming, is that it is very fun to see the result. When you change code you will get a visual feedback directly (often at least) which is very fun.

## 7 Future work

Major area of improvement is parameter tweaking. Even if I have put several hours here, you could always find something better. The color is also on thing that I would want to look a little bit more closer at.

One thing I really would have done, and I hopefully will do in the future, is to take this to 3D by using ray marching. By taking it to 3D I hope that it will make the animation look even more realistic than it does today.

## References

[1] WebGL Tutorials `http://learningwebgl.com/blog/?page_id=1217`

[2] Stefan Gustavson. *Simplex noise demystified.* 2005. `http://webstaff.itn.liu.se/~stegu/TNM084-2014/simplexnoise.pdf`

[3] Short about Flow Noise `http://webstaff.itn.liu.se/~stegu/aqsis/flownoisedemo/README.txt`

[4] Procedural Textures in GLSL `http://webstaff.itn.liu.se/~stegu/TNM084-2014/proceduraltextures.pdf`

[5] WebGL Fundementals, clipspace `http://www.html5rocks.com/en/tutorials/webgl/webgl_fundamentals/`

[6] Fire in HLSL, algorithmic how to generate fire `http://www.rastertek.com/dx10tut33.html`

[7] WebGL Reference `https://www.khronos.org/files/webgl/webgl-reference-card-1_0.pdf`

[8] David S. Ebert, F. Kenton Musgrave, Darwyn Peachey,Ken Perlin, and Steven Worley. *Texturing and Modeling: A Procedural Approach.* Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 3rd edition, 2002.