Suggested exercise for the procedural shading lab

Stefan Gustavson, stefan.gustavson@liu.se, 2017-02-06

You may do anything you like, and please feel free to experiment, but the steps below are a useful progression to get your feet wet with shader programming.
This example is short on purpose. When you're done, create something on your own!
And remember: ***Don't be afraid to ask questions!***

The web page for this exercise can be reached from anywhere:
`http://www.itn.liu.se/~stegu76/campusveckan2017/`

Draw a circle by computing the distance to the centre by `length(st-0.5)` (how does this work?), deciding with a `step()` function where the edge of the circle should be, and setting the color of the pixel by using the `mix()` function and two colors specified as two `vec4` values:

```
float radius = length(st-0.5);
float sun = step(0.3, radius);
vec4 suncolor = vec4(1.0, 1.0, 1.0, 1.0);
vec4 skycolor = vec4(0.0, 0.0, 0.0, 1.0);
gl_FragColor = mix(suncolor, skycolor, sun);
```

Relace the `step()` with a `filterstep()` to create a nice, smooth edge for the circle, and change the colors to something nicer, like yellow and dark blue:

```
float radius = length(st-0.5);
float sun = filterstep(0.3, radius);
vec4 suncolor = vec4(1.0, 1.0, 0.0, 1.0);
vec4 skycolor = vec4(0.0, 0.0, 0.2, 1.0);
gl_FragColor = mix(suncolor, skycolor, sun);
```

Add some noise to the distance to the centre, to make the circle wobbly around the edges:

```
float radius = length(st-0.5);
radius = radius + 0.04*perlinnoise(st*10.0);
float sun = filterstep(0.3, radius);
vec4 suncolor = vec4(1.0, 1.0, 0.0, 1.0);
vec4 skycolor = vec4(0.0, 0.0, 0.2, 1.0);
gl_FragColor = mix(suncolor, skycolor, sun);
```

Try changing the values `0.02` and `10.0` and see what their roles are!

Make the circle change its shape over time by adding the variable `time` to the noise function:

```
radius = radius + 0.02*perlinnoise(st*10.0, time);
```

Change `time` to `2.0*time` or `0.3*time` and watch the result!

Now, add more detail to the wobbly edge by adding two more calls to `perlinnoise()`:

```
...
radius = radius + 0.04*perlinnoise(st*10.0, time);
radius = radius + 0.02*perlinnoise(st*20.0, 2.0*time);
radius = radius + 0.01*perlinnoise(st*40.0, 4.0*time);
...
```

Make the color for the ”sun” more interesting by using noise to modify the RGB color as well:

```
...
float fire = 0.4*flownoise(st*10.0, time);
vec4 suncolor = vec4(1.0, 0.7+fire, 0.0, 1.0);
...
```

Last, make the pattern more detailed by adding another component of noise. Our use of `flownoise()` makes it possible to use the derivative of the large spots pattern to push the small spots into the valleys of the large-scale pattern, so to speak. The exact details on how this works require quite a lot of math beyond what you have studied this far, but if you pick an engineering programme for your university studies, you will learn that math in your first year.

```
...
vec2 g;
float fire = flownoise(st*10.0, time, g);
fire = fire + 0.2*flownoise(st*20.0 + 0.05*g, time);
...
```

Looking at what we have, the display doesn’t really look very much like a sun. It’s just a flat 2D image, and it’s not very detailed. If you look at a real sun, it has a lot of small detail, and it’s a sphere where the pattern bends across a curved surface. However, more detail can be added by using more components of noise with a smaller size, and a procedural pattern can be extended to 3D without much trouble, and be used to create a pattern also on a 3D object.

This is where procedural patterns become really fun and very useful. We will not get into how to put this kind of patterns on 3D objects in this short introduction. However, 3-D graphics is taught already in the first and second years of the MT programme, and there are advanced courses on this subject (among many others) towards the end of the 5-year curriculum.

If you want to explore procedural shaders further on your own, we recommend this free, web-based book with interactive examples:

`http://thebookofshaders.com/`

We helped write a couple of chapters in that book, but most of the text is written by an artist who got into programming because he was fascinated by procedural patterns. It’s not an easy read, but it’s interactive and fun!

```
uniform float time;    // Time in seconds
uniform sampler2D tex; // Texture image
varying vec2 st;       // Texture coordinates

void main() {
  float radius = length(st-0.5);
  radius = radius + 0.04*perlinnoise(st*10.0, 1.0*time);
  radius = radius + 0.02*perlinnoise(st*20.0, 2.0*time);
  radius = radius + 0.01*perlinnoise(st*40.0, 4.0*time);
  float sun = filterstep(0.3, radius);

  vec2 g;
  float fire = 0.4*flownoise(st*10.0, time, g);
  fire = fire + 0.2*flownoise(st*20.0+0.05*g, time);

  vec4 suncolor = vec4(1.0, 0.7+fire, 0.0, 1.0);
  vec4 skycolor = vec4(0.0, 0.0, 0.2, 1.0);
  gl_FragColor = mix(suncolor, skycolor, sun);
}
```