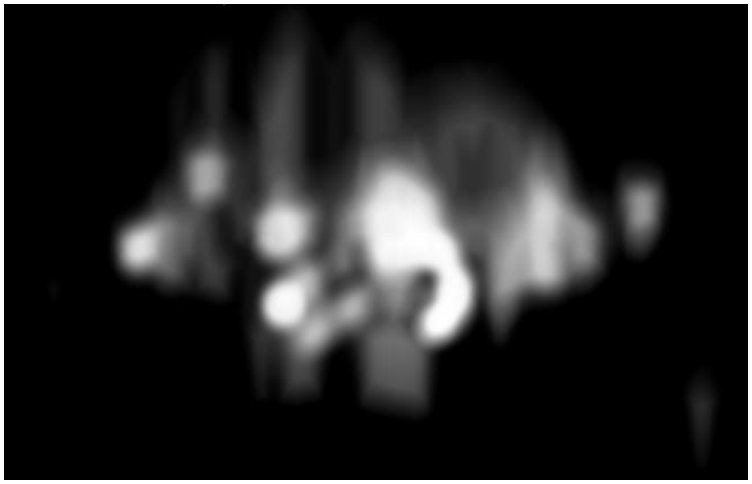
Topic: Smoke simulation

How the technique works:
Like many physical phenomena, it's a chaotic system, which makes it very difficult to predict. The state of the simulation depends heavily on the interactions between its individual particles.
Smoke simulation can tackle with the GPU: it can be broken down to the behavior of a single particle, repeated simultaneously millions of times in different locations.



This is what we need to implement to get a realistic-looking smoke effect: the value in each cell dissipates to all its neighboring cells on each iteration. Like figure 4. General technique is to have each pixel give away some of its color value, every frame, to its neighboring pixels.

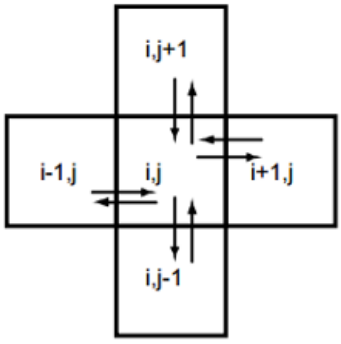
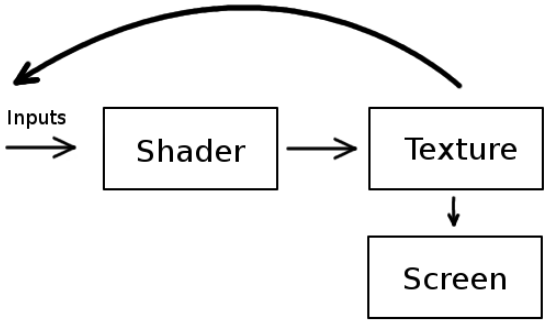


Figure 4: Through diffusion each cell exchanges density with its direct neighbors.

Main steps for GLSL shaders:
1: Moving Values Across Pixels
Each pixel should gain some color from its neighbors, while losing some of its own. We pass our uniform variables and texture to the shader, it makes the pixels slightly redder, draws that to the screen, and then starts over from scratch again.



2: Getting a Smoke Source

What our shader looks like:

```
//The width and height of our screen
uniform vec2 res;
//Our input texture
uniform sampler2D bufferTexture;
//The x,y are the posiiton. The z is the power/density
uniform vec3 smokeSource;

void main() {
	vec2 pixel = gl_FragCoord.xy / res.xy;
	gl_FragColor = texture2D( bufferTexture, pixel );

	//Get the distance of the current pixel from the smoke source
	float dist = distance(smokeSource.xy, gl_FragCoord.xy);
	//Generate smoke when mouse is pressed
	if(smokeSource.z > 0.0 && dist < 15.0){
		gl_FragColor.rgb += smokeSource.z;
	}
}
```

Result by now:

3: Diffuse the Smoke

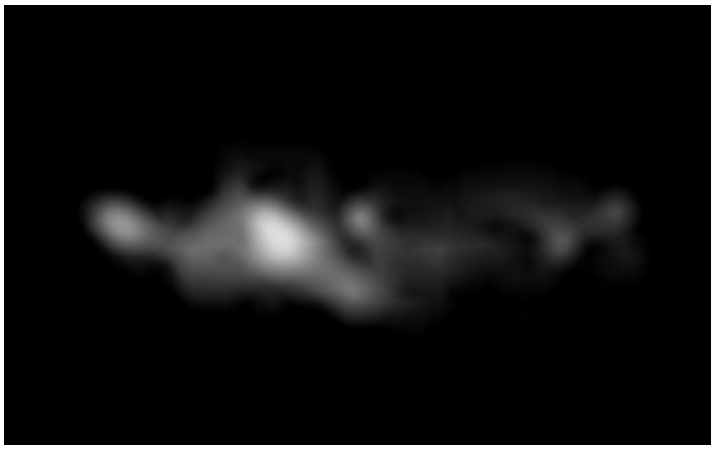
We've got all the pieces now, then finally tell the shader: each pixel should gain some color from its neighbors, while losing some of its own.

```
//Smoke diffuse
float xPixel = 1.0/res.x; //The size of a single pixel
float yPixel = 1.0/res.y;

vec4 rightColor = texture2D(bufferTexture, vec2(pixel.x+xPixel, pixel.y));
vec4 leftColor = texture2D(bufferTexture, vec2(pixel.x-xPixel, pixel.y));
vec4 upColor = texture2D(bufferTexture, vec2(pixel.x, pixel.y+yPixel));
vec4 downColor = texture2D(bufferTexture, vec2(pixel.x, pixel.y-yPixel));

//Diffuse equation
gl_FragColor.rgb +=
        14.0 * 0.016 *
        (
                leftColor.rgb +
                rightColor.rgb +
                downColor.rgb +
                upColor.rgb -
                4.0 * gl_FragColor.rgb
        );
```

Result by now:



4: Diffuse the Smoke Upwards

This doesn't look very much like smoke, though. If we want it to flow upwards instead of in every direction, we need to add some weights. If the bottom pixels always have a bigger influence than the other directions, then our pixels will seem to move up.

By playing around with the coefficients, we can arrive at something that looks pretty decent with this equation:

```
//Diffuse equation
float factor = 8.0 * 0.016 *
        (
                leftColor.r +
                rightColor.r +
                downColor.r * 3.0 +
                upColor.r -
                6.0 * gl_FragColor.r
        );
```

The final smoke result is like this, which is going upwards:

Part 2: My group members for the final project:
Jingcheng Shi, Shuli He

(Project topic is not decided.)