“Sprinkled Speech” by Chris Joseph – not yet forgotten techniques

Jim Andrews
vispo.com

“Sprinkled Speech” by Chris Joseph – not yet forgotten techniques

Jim Andrews
vispo.com

Despite its considerable capacity to delight, Chris Joseph's online, interactive, animated poem “Sprinkled Speech”[1] has hidden depths. It's easy to ignore them. As Randy Adams put it in his last comment, it has a light touch.

It has special meaning to me because, as it turns out, this is the only digital poem I can think of that Chris, myself, and Randy Adams had a hand in. Quite asynchronously, across time. Even past death. Randy passed away April 25, 2014 after an illness. He and I were old friends but were estranged, and had been since 2005. I didn't know he was sick until after he passed away. But he and Chris worked together on remixworx.com for years, up until his death. And the three of us met on several occasions.

The first time I saw “Sprinkled Speech” was early October 2017 on Chris's remarkable web site. A few weeks ago. I didn't know Randy had any involvement in the piece until I looked at the source code, which I don't tend to do unless I'm taken with a piece, and only then if I have some specific curiosity about how it works under the hood. So I'd been looking at the first version (the second version didn't exist yet) for a few days by the time I looked at the source code, which contains the following comment:

> <!--
> by babel 12/3/14, for remixworx, http://runran.net/remix_runran
> from forgotten techniques, http://runran.net/remix_runran/?p=388
> + sketch.js, https://github.com/soulwire/sketch.js
> -->

“babel” is one of Chris's aliases. I wasn't sure if the date meant December 3, 2014 or March 12, 2014. The former is several months after Randy died. The latter is just a bit longer than a month before he passed away. Also, the runran.net URLs no longer work, so I wasn't quite sure what sort of involvement Randy had in the piece.

I had emailed his wife about possibly helping maintain his site after I heard he died, but didn't hear back from her. Runran.net stayed up for a while after his death and then went

1 The original version of “Sprinkled Speech” was published by Chris Joseph on March 12, 2014 at https://remixworx.com/babel/2014/sprinkledspeech.html. A second version was published October 17, 2017, at http://remixworx.com/cj/2017/sprinkledspeech2.html .

down. But Chris was responsible for keeping the group project site http://remixworx.com running, which Randy had initiated years earlier. He and Chris, Christine Wilks, and less regular contributors such as Andy Campbell and Jukka-Pekka Kervinen, had been posting work there for years.

I emailed Chris Joseph and we corresponded about “Sprinkled Speech” for a couple of weeks as I came to know it better and, finally, write several drafts of this article, as my knowledge deepened about the piece and Randy's part in it. I've revised this several times as Chris helped me with the timing of events. I enjoyed the piece before I knew Randy had any involvement in it. When I learned he had a hand in it, it was like a surprise Randy himself would have enjoyed as an art prank. Especially if he was dead when he did it. One more time with the three of us.

It turns out that 12/3/14 refers to March 12, 2014, and Randy saw and commented on “Sprinkled Speech” within days of when Chris published it also on remixworx.com on March 12. He must have been quite ill at that point; Chris tells me this was his last comment on remixworx.com. The time of the comment was 3:52 a.m.

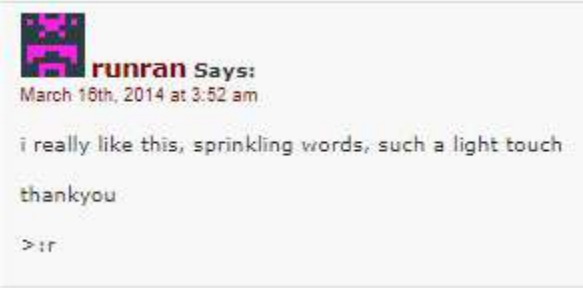

Illustration 1: Randy's last comment on his site remixworx.com.

It isn't a deep comment. He may have had other things on his mind, at that point, being so near the end. There is hardly anything about Randy's comment—other than it being his last—or “Sprinkled Speech” itself—that strongly indicates hidden depths, at first glance.

Chris's comment in the source code of “Sprinkled Speech” also references sketch.js, a JavaScript, <canvas> library that Chris used to help him create “Sprinkled Speech”. Sketch.js describes itself as a “Cross-Platform JavaScript Creative Coding Framework”. Many of the examples of the uses to which people have put sketch.js[2] show it as a handy particle system. That is, it's useful for generating a bunch of particles that can 'live' and 'die', be assigned properties and move around the screen. In our case, the particles are words. I don't see any examples in the sketch.js “showcase” of any language pieces. But that's typical of digital poetry—if often uses technologies that weren't meant for use with language but work surprisingly well with it, in some ways.

If you run the poem, you see that moving the mouse over the screen causes an incandescence of language to erupt. It is a bit like the 'sparklers' of Halloween. Or fairy dust. Which is more likely to be 'sprinkled' than Halloween sparklers, I suppose. The title is not 'Sprinkled Text' or 'Sprinkled Words', but “Sprinkled Speech”. Which is better than 'Sprinkled Text' or 'Sprinkled Words' not only because of the assonance but because of the way it makes us associate the animated text with a person's speech, with a person speaking.

The text, as is noted in the dialog box that opens should we click the “i” at bottom right of

2 See https://github.com/soulwire/sketch.js

the recent, second version of the poem, is a poem by Randy Adams called “forgotten techniques”:

forgotten techniques
sprinkled speech in special inkwells
revealed sometimes versions later
in a virgin instant, in a sacred city

desire the boundary of letters
codes, messages and memory
or save time, toss the dice

sacrifice the white sheet
believe in invisible examples
voice disorder, square balloon

As noted at remixworx.com, Randy wrote this in 2008, long before Chris created “Sprinkled Speech” in 2014. And we see that the title of Chris's piece is from the first stanza of the poem.

The title of the poem, “forgotten techniques” reads, to me, like it's set in the future, a future in which the browser and sketch.js are ancient artifacts long ago forgotten. Forgotten techniques of digital poetry. Even as another possible temporal reference point is not the future but the past, where real inkwells (as of olde) contain ink that has almost microscopic sprinkles in it, sprinkles that, like invisible ink, do not reveal themselves for some time.

But “versions later”? That does not really work in the past. It seems to me that “versions later” launches the piece onto the web. And now there are two versions of “Sprinkled Speech”.

As though, as he was living in 2008, Randy rose above time, without knowing it, via the not so forgotten techniques of poetry, into the empyrean, the timeless, the eternal. The poem's temporality is probably several.

Chris's “Sprinkled Speech” is remarkable in its interactive animation of language. The sense it gives off of joy and wonder, of beauty and passion, is unmistakable, palpable, vitally present. It's this that really tips you off to its hidden depths. The articulation of such joy is not trivial. Or, to look at it another way, Paul Celan remarked that “Attention is the soul's prayer.” That animation is so attentive.

It's almost impossible to read Randy's text. But not completely. If you click to stop the animation—being able to stop motion with a click is the new feature of the second version—you see an intriguing visual poem. The order of the words, from left to right, is the same as the order of the words in Randy's poem. A given word always occurs in the same horizontal location. For instance, the word “sacred”, which appears once in the poem, is the 16th word of 44 in the poem, so it always occurs 16/44 of the way across the width of the screen. And nowhere else. So it sort of is possible to read Randy's poem while interacting with Chris's piece—if you mouse across the entire horizontal length of the screen quickly. And then click at the right edge. A few words may be missing, though. For instance, “special” has already evanesced, in the below screenshot. And there are usually many repetitions of any given word.

Illustration 2: After having moved the mouse quickly from the left to right edge.

In the source code, we see that the language of the poem is coded in an array named "words":

```
words=["sprinkled","speech","in","special","inkwells","revea
led","sometimes","versions","later","in","a","virgin","insta
nt,","in","a","sacred","city","desire","the","boundary","of"
,"letters","codes,","messages","and","memory","or","save","t
ime,","toss","the","dice","sacrifice","the","white","sheet",
"believe","in","invisible","examples","voice","disorder,","s
quare","balloon"];
```

There are only two other mentions in the source code of this array, both of them in a function named `draw`, which is a method of the `Particle` object. `Particle` objects have three methods: `init`, `move`, and `draw`. In other words, particles can do three things: get initialized when they are born, move and then be drawn each frame of the animation (when they are 'alive').

```
draw: function( ctx ) {
       ctx.beginPath();
       ctx.fillStyle = this.color;
       ctx.font=((Math.floor(Math.random()*6)+1)/4)+'em courier
       new,courier';
       whichword=Math.floor((words.length*this.x)/window.innerWidth);
       if (whichword<0){
             whichword=0;
       }
       textString=words[whichword];
       textWidth = ctx.measureText(textString).width;
       ctx.fillText(textString,this.x,this.y);
}
```

The line that begins with `ctx.font` assigns a font and font size to a word. The font size is a random number between 1/4em and 7/4em. That's what's responsible for the motion within a word. As it moves, it randomly changes font size. Little explosions of internal movement simulate the little explosions in sparklers. That's pretty detailed textual animation of verisimilitude for a poet.

The math of what determines the font size is a bit like rolling the dice, because it's

random, and we see `Math.random()*6`. Dice have six sides. This reminds me of the second stanza of Randy's poem:

desire the boundary of letters
codes, messages and memory
or save time, toss the dice

In this case, it doesn't "save time". It gives the words a little explosive meaning beyond the lexical; the toss of the dice makes the word randomly change size, giving it a little explosive behavior.

The line that starts with `whichword` assigns a particle a word. The word assigned to a particle depends on `this.x`, which is the horizontal component of the particle's position. The `words` array doesn't change at all throughout the poem, so `words.length` is always 44, the number of words in the poem. The only time `window.innerWidth` changes is if the reader resizes the browser window. So `whichword` depends on `this.x`, the horizontal position of the particle. `this.x/window.innerWidth` is always a number between 0 and 1 because `this.x` is always 0 or greater, and is always less than `window.innerWidth`, which is the width of the browser window.

Consequently, any given word always appears in the same horizontal area. Although the poem is in many ways non-linear, it is linear in this sense. A word that appears half-way through the poem only appears half way across the screen. A word that appears at the beginning of the poem only appears at far left of the screen. And so on.

The line that starts with `ctx.fillStyle` shows us that the color of a particle remains constant over its entire life; the color is fetched, not computed. And we see in `demo.spawn`, where particles are generated/birthed, that a particle is assigned a random color from the seven-element `COLOURS` array at birth.

However, that doesn't quite line up with our experience of the poem. We see that, very close to the mouse (or your finger, if you run the poem on a mobile device), the words are numerous and they're whitish in color. But as they move further away from the mouse, the color changes and seems to stay the same thereafter. How does this line up with the fact that once the color of the text is set, it doesn't change in the code? The answer to this question shows Chris Joseph operating as a visual artist of considerable subtlety and skill as a programmer.

We see a method of the app called `demo.draw`; it's the program's main drawing method. It iterates over all particles, drawing them each frame of the animation. Here it is:

```
demo.draw = function() {

        demo.globalCompositeOperation  = 'lighter';

        for ( var i = particles.length - 1; i >= 0; i-- ) {
                particles[i].draw( demo );
        }
};
```

It's the line that sets `demo.globalCompositeOperation` to `'lighter'` that makes the text white near the cursor and a different color when the text moves away from the cursor. The value of `globalCompositeOperation` is an important JavaScript setting that determines how the colors of layers within a canvas are composited. As we see at

mozilla.org, `'lighter'` makes the color of overlapping things be lighter. Whiter, that is. The more things that overlap, the lighter/whiter the resulting color. The fewer things that overlap, the closer the colors are to whatever they are set to by the code. This turns out to be a brilliant simulation of white heat, of combustion, in the case of "Sprinkled Speech". The text is spawned and close together near the cursor, so more nearly white there than elsewhere.

This is one of the most artistic uses of `globalCompositeOperation` I've seen. How artists use this, if at all, can tell you about them as artists, and especially as colourists. Chris Joseph knows `globalCompositeOperation` inside and out. You can see it in his other works on chrisjoseph.org such as "Amy Muller" and several other works in which he experiments with the effects it has in various situations.

And he created the graphics he gave me for Aleph Null 3.0[3] by compositing pairs of images with `globalCompositeOperation='difference'`. I would have thought it was done using Photoshop for hours. Instead, he used JavaScript and, from 37 images, composited them pairwise with `globalCompositeOperation='difference'` and took the best results of that process. That's quite a few possibilities to check out. He noted to me that the number of combinations to check out is 666. Yes indeed: 37x36/2=666. Coincidence? Well, yes, but a humorous one.

The code assigns words any one of seven colors coded in the `COLOURS` array:

```
var COLOURS=
['#69D2E7','#A7DBD8','#E0E4CC','#F38630','#FA6900',
'#FF4E50', '#F9D423'];
```

We see those colours are these:

Illustration 3: The colours of the text in "Sprinkled Speech".

These colors are a bit different from the colours in sparklers. We see quite a bit of the bottom five in sparklers. But not so much of the top two bluish ones. But this set is also more *readable* than the colours of sparklers, which are sufficiently close together in colour to make reading difficult, if the words are close together. The above set contains

3 Chris's contribution to Aleph Null 3.0 is at http://vispo.com/aleph3/an.html?d=Chris%20Joseph

more contrast. There is enough verisimilitude to be delightful, especially in the motion of the little explosions and the white heat of the combustion. Readability is important too.

I first encountered this poem only recently. Chris does not announce much of his new work, so if you don't get over to his site and check out what's new, you can miss a lot of quality work. It was just a couple of weeks ago that I encountered the first version of the poem . There was no feature whereby, if you click the canvas, that toggles the animation. That's the biggest difference between the first and second versions. And there was no second version. However, I noticed by accident that, in the first version, if you move the mouse off of the browser and then click, the animation pauses, and if you then subsequently click on the canvas, the animation resumes.

In the meantime, I'd sent the poem to an old friend of mine: the poet Derk Wynand. Derk wrote back saying he thought it was technically accomplished, but he wasn't sure how much of it “stuck”. If you can't stop the animation, that's a perfectly understandable response. If you can't stop the animation, you can't really read it very well.

But once you can stop and start the animation, things change. Suddenly, when the poem is stopped, you have a strong visual poem and, additionally, a fascinating text. The fascination of the text is initially just in figuring out what is *going on* with this text. We see repetitions of any given word. But we don't see a given word repeated in different parts of the screen; all the repetitions of a word are isolated to a particular horizontal area. Eventually we see that the poem can be read left to right, if we eliminate repetitions.

I emailed Chris about my delight with this poem. We exchanged a few missives. And I noted that being able to stop and start the animation adds significantly to the poem. Mostly it's a matter of adding to the readability of the poem. It's nearly impossible to appreciate the text and also the way the vocabulary of the poem is apportioned out across the screen without being able to stop and start the animation. It becomes much more *literary* when you can stop the animation and read it as a striking visual poem.

Anyone old enough to have experienced the web in the 1990's remembers the mouse trail decorations that you might often find on web pages. You'd visit somebody's personal site, and they'd have some gawd-awful JavaScript widget installed that would chase the mouse around the screen. “Sprinkled Speech”, even without the ability to stop and start the animation, is leagues beyond those pieces.

But with the added ability to stop and start the animation, a new dimension is introduced beyond being an interactive, highly charming animation. It operates, as a poem, in four ways: as a static text spread out horizontally, with repetitions of words; it also operates as a striking visual poem of heated word clouds; and as an animated/kinetic, interactive poem; and, given that the second version includes a link to the text of the poem layed out as a poemy poem, we also consider it as that sort of poem.

Finally, I think I wrote Chris an email that described how I would change the piece from the initial version, the basically only change being the ability to stop and start the animation. I'd seen his other works often had an 'i' at bottom right which, when clicked, open up a dialog box with information about the poem. That would note that clicking the screen toggled the animation on and off. There's a good argument not to put such stuff in a piece because it's like explaining a joke. But it's simple enough to document.

I had barely sent that email to Chris—he wouldn't have even had time to read it—when I received a reply from him saying that he had created a new version. And this was exactly as I thought it would be—only better. The dialog box also notes that the original text was

by Randy, and supplies a link to the original text. That is when I first read Randy's text.

Reading that text really gave my head a spin. Which I'm sure Randy would be happy about. It was that “virgin instant” he described, that experience of Randy “revealed” in a later version. In a not so sacred city, I must say. It's a poem about the nature of his life's work. And mine. And Chris's. In which we “toss the dice” to arrange some things. But the “boundary of letters/codes, messages and memorys” requires a deeper arrangement, sometimes, a different arrangement than the dice can provide. We “sacrifice” the blank page—and lots of them, trying many things so that what we finally use/keep leaves “invisible examples”.

As I said, Randy and I were estranged for many years. But there was a time when we shared a belief in this new literature, this new art, and worked together. The three of us still share quite a bit. Randy so recently, and temporarily, back from the dead in this poem. None of us are/were novelists or story writers. It's the brilliant moment we seek. He really got me with this one. In that moment of recognition of him seemingly beyond time, stepping out from behind the curtain in Chris's amazing poem.