The Breast Cancer Twibbon Tree

An Interactive Twitter Poetry Project

Artist's Statement

Lisa (Biqing) Li
LMC 6310 Computer as Expressive Medium

Project Overview

October is Breast Cancer Awareness Month, an annual international health campaign dedicated to raise consciousness and funds for women diagnosed with the disease. The campaigns offers support and information to those whose lives are affected by breast cancer.

Breast cancer is the second most common kind of cancer in women. Every 1 in 8 women born today in the United States will get breast cancer at some point in her life. The good news is that many women can survive breast cancer if treated early. Unfortunately, many women also did not make it. In my Twitter-Processing poetry project, I want to dedicate this project to augment the ongoing social media presence for breast cancer awareness, to show support for breast cancer survivors, and pay respect to those who did not make it.

Aesthetic Choice

Pink Sky with Dimming Effect

I borrowed the dimming effect idea from “Relay for Life”, an event sponsored by the American Cancer Society that gives people a chance to celebrate the lives of people who have battled cancer, remember loved ones lost, and fight back the disease. At the relay, participants team up and camp out at a local ground, taking turns walking or running around a track or path. Each team must have a representative on the track at all times during the event. Cancer never sleeps, therefore the relay goes on throughout the night, from dusk to dawn, for 24 hours. Therefore, the background of the application constantly dims down from a pink-dark pink gradient to a pink-black gradient, symbolizing cancer’s encroachment on the patients’ health. Also, it represents the long, dark journey battle cancer patients experience.

Twinkling Stars
Each star in the sky corresponds to an individual tweet by a Twitter user who tweets with the hashtag #breastcancer. When the user hovers over individual stars, the tweet text and the user’s profile image show up at the cursor’s location.

Inspirational Word Cloud
To enhance the poetic expression of the program, I have a pre-generated list of breast-cancer-related inspirational phrases, such as “fearless”, “healing”, “survivor”, “pink”, “ribbon”, etc. I cross-compare this list with the words in the live tweets. Whenever there are overlaps, the inspirational words appear on the canvas in big, white, semitransparent fonts. As the sky dims, the user can bring daylight back to the sky by hovering over the stars and finding these “inspirational words”.

Shooting stars
Shooting stars have always been revered with awe and amazement throughout history. We have a ritual of making a wish upon seeing a shooting star in the hope that divinity will grant our request. Seeing a shooting star stands for good luck, hope, and aspiration for future. We share the same awe and respect for the divine forces whenever we wish upon a star. In the program, shooting stars constantly appear and disappear in the night sky backdrop. The shooting stars are motifs dedicate to the souls lost to breast cancer. These streaking lights symbolize the eternity of life, the wonder of our existence, and the vastness we have yet to discover.

Tree of life motif
The tree of life is a mystical and magical symbol in many cultures. I introduced the concept of tree of life in the program, which is a common motif to alludes to the interconnection of all lives on earth. Pink ribbons in various sizes scattered on the branches like tree leaves, representing lives lost to breast cancer but remain everlasting in the memories of people.

Pink Cursor
I customized the cursor design of the program to resonate with the pink breast cancer theme.

Algorithmic Choice

User-Defined Class
I built a class named "**ShootingStar**" where I define the animation and behaviors of each shooting star appearing on the screen. Inside the class, there are 5 functions: `void run()`, `void move()`, `void fade()`, `void drawShootingStar()`, `newShootingStar()` that control the movement of the shooting star. Basically, whenever one shooting star fades out, it triggers the next shooting star to appear on the canvas. I declare an instance "`myShootingStar`" and call the functionality inside of my `draw()` loop.

Dimming Effect
Because I want to represent the "daytime-nighttime-daytime" transition, I manipulated the background gradient so that the gradient gradually stretches out to simulate the changing process of lighting. The gradient goes from black to pink from top down. To draw the gradient, I use a function `setGradient()`:

```
setGradient(0, 0, width, height+numOfFrame, b2, b1, Y_AXIS)
```

I set the variable `numOfFrame` to be `numOfFrame+=2` or `numOfFrame -=2`
depending on a Boolean value `dimming`. When `dimming` is `true`, the gradient stretches out in height frame by frame, Whenever I hover over a star, dimming is set to false, and the background gradually lights back up.

Stars and Live Tweets

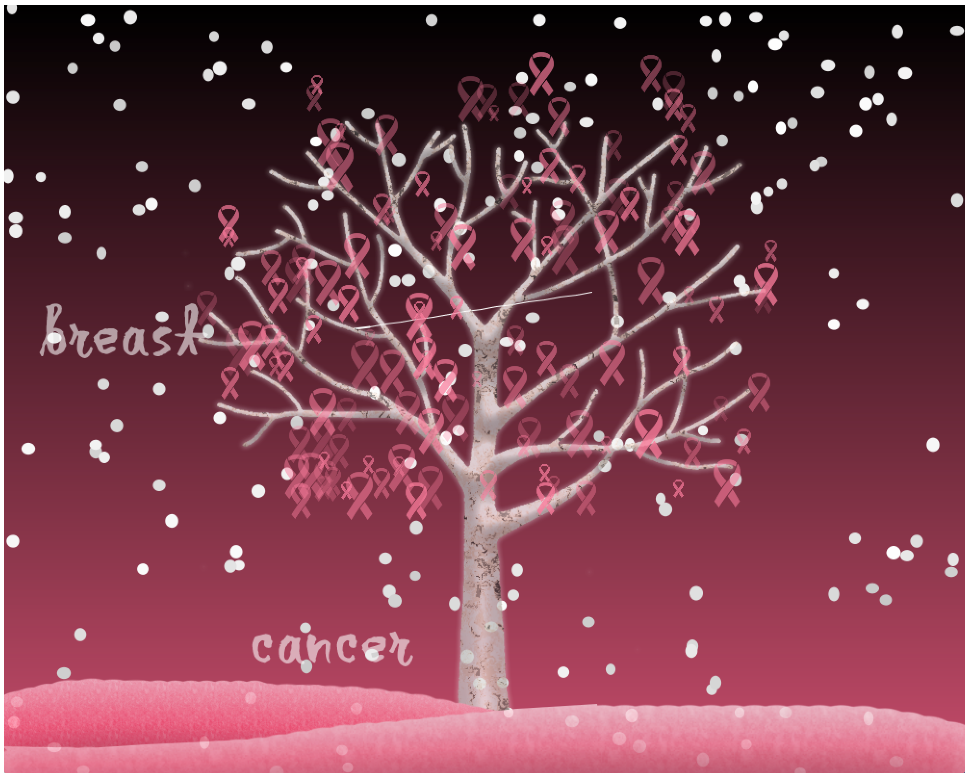
For individual tweet texts on display, I cleaned up the texts by getting rid of "-RT", twitter handles (@), and replacing hashtags with spaces (" "). To populate a repository of tweets, I searched "#breastcancer -RT" using Twitter API and stored them inside `tweetText` array after they are cleaned up. Each star corresponds to a Twitter's tweet talking about #breastcancer. The stars are randomly generated and sprinkled all over the canvas using:

```
starX[i] =(int)random(width);
starY[i] = (int)random(height);
```

The user can change the size of the stars by changing the value of "starSize" on line 119:

```
int starSize = 5; // the size of the stars
```

An example of a bigger `starSize` is displayed as follow:



starSize = 20

The tweets and profile pictures are pulled live from Twitter and displayed on the canvas using:

```
for (int k=0; k<starX.length-1; k++) {
      if (dist(mouseX, mouseY, starX[k], starY[k]) < starSize*2) {
        dimming=false;
        String tweetText;
        if (k<modifiedtweets.size()) {
          tweetText=modifiedtweets.get(k);

        } else {
          tweetText=modifiedtweets.get(k%modifiedtweets.size());

        }
        fill(214,181,196,255);
        textLeading(1);
        tint(255);
        textFont(font2);
        textAlign(CENTER, BOTTOM);
        tint(132, 11, 76,100);

        text(tweetText, starX[k],starY[k]);

        if(k>=100){
     image(ProfileImages.get(k
%modifiedtweets.size()),starX[k],starY[k]);
        }
        else{
          image(ProfileImages.get(k),starX[k],starY[k]);
        }
        //println(k+":"+tweetText);
       // noLoop();
       tint(255);
     }
    }
   }
  }
```

However, there is a bug with the line:

```
text(tweetText, starX[k],starY[k])
```

(highlighted in red) that I could not wrap the tweet texts inside a box. It would be idea if the tweet texts do not show up in one long line and go off the canvas, but they simply would not show up if I put an arbitrary box around them. So I left them as is.

Inspirational Phrases

To display the individual phrases on the screen, I use another random generator to populate the positions for these phrases:

```
    wordX[i] =(int)random(100,800);
    wordY[i] = (int)random(100,600);
```

To decide whether to display a word or not, I have to check whether any words in the list overlap with the tweet tex. To do so, I first created a array of Boolean values called `wordVisibility`:

```
    boolean[] wordVisibility;
    wordVisibility=new boolean[numOfWords];
```

I then created an array of floats to hold the transparency values of the inspirational words on display:

```
    float[] wordTransparency;
```

To test whether each word in the text document should be displayed or not, the program checks the Boolean values each time the tweets are pulled and stored inside `tweetText`, as mentioned previously. For example, if a tweet contains the word "empower", which is a phrase inside the "inspirational word" list, the corresponding Boolean value for "empower" will be changed to "true", which triggers the wordTransparency value to increase by 1.

```
    for (int word=0; word<inspirationalWords.length; word++) {
        if (tweetText.contains(" " + inspirationalWords[word] + " "))
          {
            //println(inspirationalWords[word]);
            wordVisibility[word] = true;
            wordTransparency[word] += 1;

            if (wordTransparency[word] > 255) {
              wordTransparency[word]=255;
            }
          }
        }
```

Otherwise, if the user is not hovering the cursor over a star, the function automatically decreases the transparency of inspirational phrases (by 0.5 each frame):

```
for (int i=0; i< wordVisibility.length; i++) {

    if (wordTransparency[i]<0) {
      wordTransparency[i] = 0;
    }

    if (wordVisibility[i]==true) {
      wordTransparency[i] -= 0.5;

      fill(255, 255, 255, wordTransparency[i]);
        //if ((wordX[i] < 200 && wordY[i]<800) || (wordX[i] > 800 &&
wordY[i]<800) ) {
        textFont(font1);
          text(inspirationalWords[i], wordX[i]+20, wordY[i]-20, 120,
80);
}
  }//for
}
```

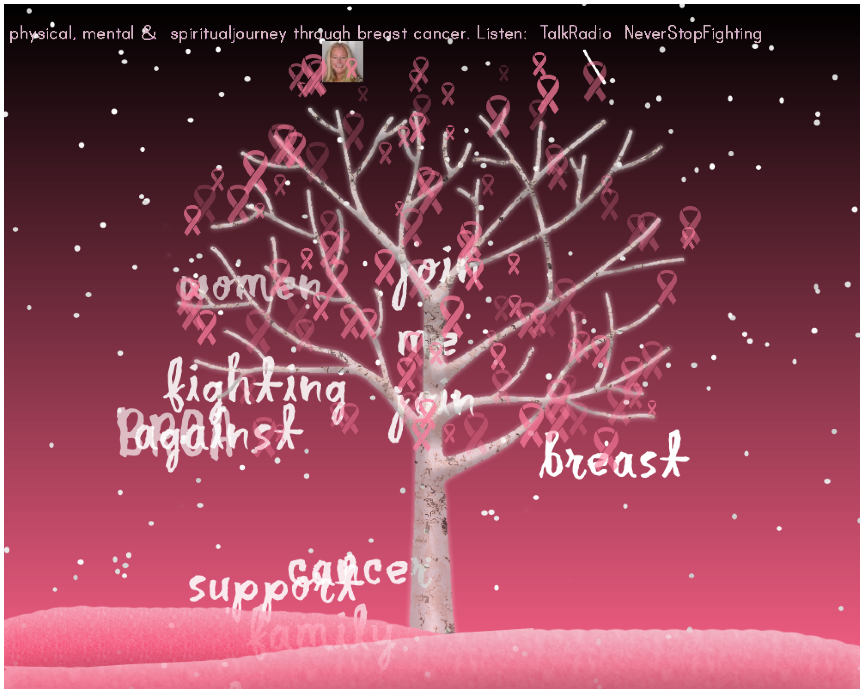
As a result, when hovering over a star, if the tweet contains any words that overlap with the list of inspirational words, the overlapped phrases will appear on the screen. If the user moves the mouse away, the phrases gradually fade away unless the user finds another star (tweet) that contains the same phrase to bring up the transparency again. Augmented by the continuous dimming effect of the backdrop, the fading phrases add to the poetic expression and interactive fun of the program, since the user has to actively move he cursor to different stars in search of tweets that contain these pre-selected inspirational words to bring up the background lighting and keep the inspirational phrases on the canvas.

Pink Ribbon on Tree of life

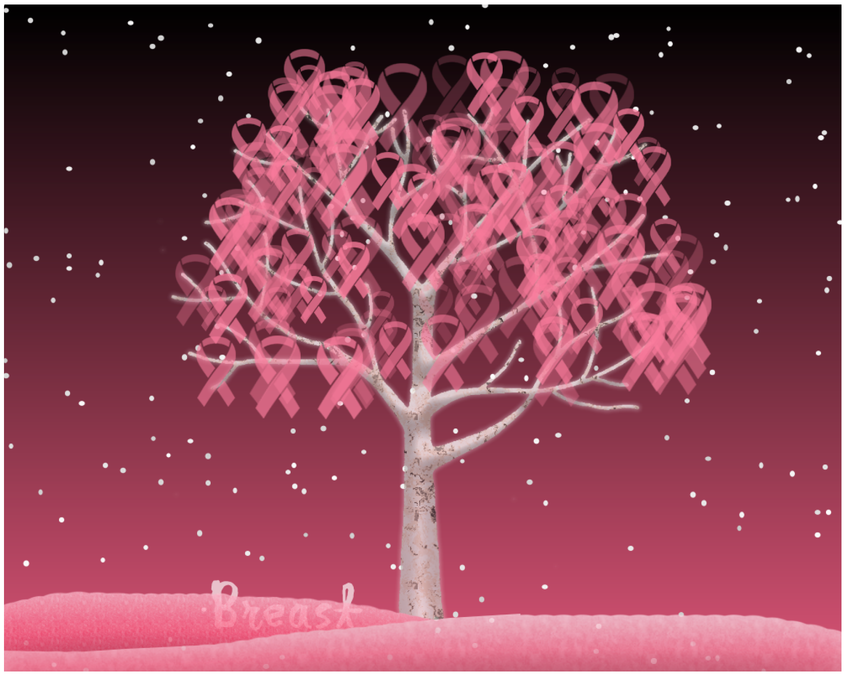
The tree trunk and the pink ribbons are images imported into Processing. To represent leaf-and-tree effect, the ribbons are randomly scattered on the tree with random sizes and transparency, within a radius of 300 from the center point of the canvas. The ribbons always appear within a circular radius of the center of the trunk to give the tree crown a natural shape. The semitransparent ribbons overlap on top of each other to simulate actual leaves and tree. Since they are randomly generated, each time the program loads, the tree crown looks slightly different. the user can alter the number and size of ribbons according to his/her discretion by changing the following values on line 107-112:

```
int ribbonXMin=150;
int ribbonXMax=850;
int ribbonYMin=50;
int ribbonYMax=500;
int ribbonSizeMin=10;
int ribbonSizeMax=30;
```

An example of different visualizations after I adjusted ribbon size and placements is shown as below:


physical, mental & spiritualjourney through breast cancer. Listen: TalkRadio NeverStopFighting
women
join
me
fighting
against
join
breast
support
cancer
family

with more scattered placement and smaller ribbon



with more compact placement and bigger ribbon

Additional Links

Link to Portfolio Page:
http://www.lisalidesign.com/portfolio/the-twibbon-tree

Link to Video:
https://vimeo.com/143818266

Link to Design Notes:
https://docs.google.com/document/d/1907zI5AaGi-1c4aLhRoFd181D45M-nFiKguwrvRn1PI/edit?usp=sharing