README

Yingxin Wei (5517682)

1. Pseudo-random selection of characters.

The sentence I choose is “rain on the green grass but not on me”. I insert several random letters in the sentence and it becomes "rain afc on the dof green grass huj but not wmz on me". I copy the sentence for a lot of times and split the sentences to letters. The x coordinates of all letters should be within a range. So rain looks random but there is a reasonable likelihood of catching a set of raindrops that spell out a whole word.

```
String rainString = "rain afc on the dof green grass huj but not wmz on me";
drops = new Letter[count][rainString.length()];
int gap = 1280/rainString.length();
int startX = 5;
inputLetters = new char[rainString.length()];
inputLetters = splitString(rainString);
int startYDiff = 0;
for (int i = 0; i < count; i++){
  for (int j = 0; j < inputLetters.length; j++){
    Letter dropLetter = new Letter(inputLetters[j]);
    dropLetter.x = startX;
    dropLetter.y -= startYDiff;
    drops[i][j] = dropLetter;
    startX += gap;
    startX += int(random(-10,10));
    if (startX >= inputImage.width-gap){
      startX = 5;
    }
  }
  startYDiff += 50;
}
```

Figure 1: Algorithm for pseudo-random selection of characters

2. Flipping the video image

I flip the video image displayed on the screen by changing the color information between (x,y) and (Imagewidth-1-x,y).

```
// Flip the video image displayed on the screen
if (userView == true){
  for (int x = 0; x < inputImage.width; x++) {
    for (int y = 0; y < inputImage.height; y++ ) {
      int loc = x + y*inputImage.width;
      int mirror_loc = inputImage.width - x - 1 + y*inputImage.width;
      // Test the brightness against the threshold
      destination.pixels[mirror_loc]  = inputImage.pixels[loc];  // White
    }
  }
}
else {
  for (int x = 0; x < inputImage.width; x++) {
    for (int y = 0; y < inputImage.height; y++ ) {
      int loc = x + y*inputImage.width;
      int mirror_loc = inputImage.width - x - 1 + y*inputImage.width;
      // Test the brightness against the threshold
      if (brightness(inputImage.pixels[loc]) > threshold) {
        destination.pixels[mirror_loc]  = color(255);  // White
      }  else {
        destination.pixels[mirror_loc]  = color(0);    // Black
      }
    }
  }
}
```

Figure 2: Algorithm for flipping the video image.

3. Control the threshold

User can change the value of the threshold used to determine foreground and background objects interactively by using the UP and DOWN arrow keys for situations with different lighting conditions. When UP is pressed, threshold increases by one; when DOWN is pressed, threshold decreases by one.

```
if (key == CODED) {
  if (keyCode == UP) {
    // up arrow key pressed
    threshold += 1;
  }
  else if (keyCode == DOWN) {
    // down arrow key pressed
    threshold -= 1;
  }
  if (threshold <=0) threshold = 1;
  else if (threshold >= 255) threshold = 254;
}
```

Figure 3: Algorithm for controlling the threshold.

4. Control the user_view

When SPACEBAR is pressed, the program will toggle between displaying the final user view and a "debugging" view that shows a binary image for the foreground/background classification, where all pixels classified as foreground are pure black and all pixels classified as background are pure white. This is accomplished by changing the variable userView when SPACEBAR is pressed and displaying the view based on the value of userView. Similarly, When "g" button is pressed, the program will toggle between the gray view and natural view. A gray filter would be added to the displayed image, which makes it looks like Utterback's program.

```
else if (key == ' ') {
  // space bar pressed
  userView = !userView;
}
else if (key == 'g') {
  grayMode = !grayMode;
}
```

Figure 4: Algorithm for controlling the user view (Part 1)

```
if (grayMode == true) {
  destination.filter(GRAY);
}
```

Figure 5: Algorithm for controlling the user view (Part 2)

5. Dropping and Rising of the Letter

The letter drops if its new assigned position doesn't belong to a dark object, that is, the brightness of the pixel is larger than the threshold. The letter rises up if the dark object moves up, that is, the pixel is less than the threshold and its above pixel is also less than the threshold. The variable "upSpeed" controls the height it rises up to.

```
for (int i = 0; i < count; i++) {
  for (int j = 0; j < inputLetters.length; j++) {
    if (drops[i][j].y < destination.height && drops[i][j].y > 0){
      //  Make rain doesn't pass through black regions even if they are very thin
      int loc = drops[i][j].x + (drops[i][j].y*destination.width);
      float brightness = brightness(destination.pixels[loc]);
      if (brightness > threshold){
        drops[i][j].dropLetter();
        drops[i][j].upSpeed = 1;
      }
      else {
        if (drops[i][j].y > 0){
          // Lift the letter if the palm of the viewer rises up
          int upLoc = loc = drops[i][j].x + (drops[i][j].y-1)*destination.width;
          float upBrightness = brightness(destination.pixels[upLoc]);
          if (upBrightness < threshold){
            drops[i][j].liftLetter();
            drops[i][j].upSpeed = drops[i][j].upSpeed*2;
          }
        }
      }
    }
    else {
      // The letter drops if it's above the screen
      drops[i][j].dropLetter();
    }
    drops[i][j].drawLetter();
  }
}
```

Figure 6: Algorithm for the dropping and rising of the letter.

7. Velocity of drops

The millis() function is related with real time and it helps control the velocity of the dropping of the letter. The program updates the image when the passed time (passedTime) is larger than the setting time (totalTime). In other words, the letter drops a pixel when the passed time (passedTime) is larger than the setting time (totalTime). So the velocity of dropping is actually

1/totalTime (pixel/ms) = 1000 /totalTime (pixel/second).

```
int passedTime = millis() - savedTime;
if (passedTime > totalTime) {
  savedTime = millis(); // Save the current time to restart the timer!
```

Figure 7: Timer for controlling the velocity of the letter.

Wizards

1. Fade away

The letter fades away if it is near the bottom of the image or if it stays in the image for a long time.

```
void letterFade() {
    alpha -= 50;
    // Letter goes to the above of the image after it fades away
    if(alpha <= 70) {
      y = int(random(-1280, 0));
      alpha = 255;
      stayTime = 0;
    }
}

void dropLetter() {
    y++;
    stayTime++;
    // Letter fades away if it is near the bottom of the image
    if (y > 550) {
        letterFade();
    }
    // Letter fades away if it stays in the image for a long time
    if ((y > 0) && (stayTime > allowTime)){
        letterFade();
    }
}
```

Figure 8: Algorithm for the vanish of the letter