```
 1: // A demonstration of text processing in the form of a
 2: // speed-reading application, somewhat akin to Spritz.
 3: //
 4: // Author: Michael Goldwasser
 5:
 6: // contstants that can be tuned
 7: int WORDS_PER_MINUTE = 1000;
 8: int FONTSIZE = 70;
 9:
10: //String sourcefile = "http://www.gutenberg.org/cache/epub/2591/pg2591.txt";
11: int NUM_SAMPLES = 9;
12: int num = int(1 + random(NUM_SAMPLES));
13: String sourcefile = "example" + num + ".txt";
14:
15: // globals used to store text and progress
16: String[] words;  // will be loaded from file
17: int cur=0;       // index of current word
18: boolean running = false;
19:
20: // global to help with font metric
21: float tickMarkX;      // x-coordinate of tick mark for placing word
22: float baseline;
23:
24: // We want canvas size to depend on FONTSIZE, but Processing 3 will not
25: // allow variable size within setup.
26: void settings() {
27:   size(13*FONTSIZE, 3*FONTSIZE);
28: }
29:
30: void setup() {
31:   textSize(FONTSIZE);
32:   noLoop();    // don't start drawing until window clicked
33:   frameRate(WORDS_PER_MINUTE/60.0);
34:
35:   background(255);
36:   strokeWeight(2);
37:
38:   tickMarkX = 0.33*width;
39:   baseline = 2*FONTSIZE - textDescent();
40:
41:   line(0, 0.1*FONTSIZE, width, 0.1*FONTSIZE);
42:   line(0, 2.9*FONTSIZE, width, 2.9*FONTSIZE);
43:   line(tickMarkX, 0.1*FONTSIZE, tickMarkX, 0.5*FONTSIZE);
44:   line(tickMarkX, 2.5*FONTSIZE, tickMarkX, 2.9*FONTSIZE);
45:
46:   loadTextFile(sourcefile);
47:   renderWord("Click to Begin");
48: }
49:
50: // used to (re)start the rendering of the prose
51: void mouseClicked() {
52:   if (words != null) {
53:     running = !running;
54:     loop();
55:     if (cur >= words.length) {
56:       cur = 0;  // start over
57:     }
58:   }
59: }
60:
61: void draw() {
62:   if (running) {
63:     renderWord(words[cur]);
64:     cur++;
65:     if (cur == words.length) {
66:       running = false;
67:       noLoop();
68:     }
69:   }
70: }
71:
```

```
72: // Display a single word to be read
73: void renderWord(String word) {
74:   noStroke();
75:   fill(255);
76:   rect(0, 0.5*FONTSIZE, width, 2*FONTSIZE);
77:
78:   // divide word into three portions (left, middle, right).
79:   int divide = getRedIndex(word.length());
80:   String left = word.substring(0, divide);
81:   String middle = word.substring(divide, 1+divide);
82:   String right = word.substring(1+divide);  // until the end
83:
84:   textAlign(CENTER);
85:   fill(255, 0, 0);          // red character
86:   text(middle, tickMarkX, baseline);
87:
88:   fill(0);                  // black characters
89:   textAlign(RIGHT);
90:   text(left, tickMarkX - 0.5 * textWidth(middle), baseline);
91:   textAlign(LEFT);
92:   text(right, tickMarkX + 0.5 * textWidth(middle), baseline);
93: }
94:
95: // Open a text file and break its text into individual words,
96: // as separated by spaces. Those words are loaded into the
97: // global 'words' array.
98: void loadTextFile(String filename) {
99:   String[] lines = loadStrings(filename);
100:
101:   // first, let's count the total number of words in document
102:   int total = 0;
103:   for (int j=0; j < lines.length; j++) {
104:     total += splitTokens(lines[j]).length;
105:   }
106:
107:   // Now let's resize global array of words and fill it
108:   words = new String[total];
109:   int w=0;  // index into words
110:   for (int j=0; j < lines.length; j++) {
111:     String[] pieces = splitTokens(lines[j]);
112:     for (int k=0; k < pieces.length; k++) {
113:       words[w] = pieces[k];
114:       w++;
115:     }
116:   }
117: }
118:
119: // Given a particular word length, return index of character that becomes red
120: int getRedIndex(int length) {
121:   return (length + 2) / 4;  // integer division
122: }
```