

```
1 class Screen {
2   static boolean color;
3   static Array bit_masks;
4
5   /** Initializes the Screen. */
6   function void init() {
7     let color = true;
8
9     let bit_masks = Array.new(16);
10    let bit_masks[0] = 1;
11    let bit_masks[1] = 2;
12    let bit_masks[2] = 4;
13    let bit_masks[3] = 8;
14    let bit_masks[4] = 16;
15    let bit_masks[5] = 32;
16    let bit_masks[6] = 64;
17    let bit_masks[7] = 128;
18    let bit_masks[8] = 256;
19    let bit_masks[9] = 512;
20    let bit_masks[10] = 1024;
21    let bit_masks[11] = 2048;
22    let bit_masks[12] = 4096;
23    let bit_masks[13] = 8192;
24    let bit_masks[14] = 16384;
25    let bit_masks[15] = 16384 + 16384;
26
27    return;
28  }
29
30  /** Erases the whole screen. */
31  function void clearScreen() {
32    do Screen.setColor(false);
33    do Screen.drawRect(0, 0, 511, 255);
34    do Screen.setColor(true);
35
36    return;
37  }
38
39  /** Sets the color to be used in further draw commands
40   * where white = false, black = true. */
41  function void setColor(boolean b) {
42    let color = b;
43
44    return;
45  }
46
47  /** Draws the (x, y) pixel. */
48  function void drawPixel(int x, int y) {
49    var int row, col, bit, address, value;
```

```
51 let col = x;
52 let row = y;
53
54 if ((col < 0) | (col > 511) | (row < 0) | (row > 255)) {
55     do Sys.error(7); // Screen.drawPixel: Illegal pixel coordinates
56 }
57
58 let bit = col - ((col / 16) * 16); // col % 16
59
60 let address = 16384 + (row * 32) + (col / 16);
61
62 let value = Memory.peek(address);
63
64 if (color) { // black
65     let value = value | bit_masks[bit];
66 } else { // white
67     let value = value & ~bit_masks[bit];
68 }
69
70 do Memory.poke(address, value);
71
72 return;
73 }
74
75 /** Draws a line from (x1, y1) to (x2, y2). */
76 function void drawLine(int x1, int y1, int x2, int y2) {
77     var int a, b, a_inc, b_inc, dx, dy, criterion, temp;
78
79     if ((x1 < 0) | (x1 > 511) | (y1 < 0) | (y1 > 255) |
80         (x2 < 0) | (x2 > 511) | (y2 < 0) | (y2 > 255)) {
81         do Sys.error(8); // Screen.drawLine: Illegal line coordinates
82     }
83
84     let a = 0;
85     let b = 0;
86
87     if (~(x1 < x2)) {
88         let temp = x1;
89         let x1 = x2;
90         let x2 = temp;
91
92         let temp = y1;
93         let y1 = y2;
94         let y2 = temp;
95     }
96
97     let dx = x2 - x1;
98     let dy = y2 - y1;
```

```
100 // do Screen.drawPixel(a, b);
101
102 if (dy < 0) {
103     let b_inc = -1;
104 } else {
105     let b_inc = 1;
106 }
107
108 if (dy = 0) {
109     while (~(a = dx)) {
110         let a = a + 1;
111         do Screen.drawPixel(x1 + a, y1);
112     }
113
114     return;
115 }
116
117 if (dx = 0) {
118     while (~(b = dy)) {
119         let b = b + b_inc;
120         do Screen.drawPixel(x1, y1 + b);
121     }
122
123     return;
124 }
125
126 while ((a < dx) & (b < dy)) {
127     let criterion = (a * dy) - (b * dx);
128
129     if (criterion < 0) {
130         let a = a + 1;
131     } else {
132         let b = b + 1;
133     }
134
135     do Screen.drawPixel(x1 + a, y1 + b);
136 }
137
138 while ((a < dx) & (b > dy)) {
139     let criterion = (a * dy) - (b * dx);
140
141     if (criterion > 0) {
142         let a = a + 1;
143     } else {
144         let b = b - 1;
145     }
146
147     do Screen.drawPixel(x1 + a, y1 + b);
148 }
149
```

```
150     return;
151 }
152
153 /** Draws a filled rectangle where the top left corner
154  * is (x1, y1) and the bottom right corner is (x2, y2). */
155 function void drawRectangle(int x1, int y1, int x2, int y2) {
156     var int temp, dx, dy, a, b;
157
158     if ((x1 < 0) | (x1 > 511) | (y1 < 0) | (y1 > 255) |
159         (x2 < 0) | (x2 > 511) | (y2 < 0) | (y2 > 255) |
160         (x1 > x2) | (y1 > y2)) {
161         do Sys.error(9); // Screen.drawRectangle: Illegal rectangle coordinates
162     }
163
164     let a = 0;
165     let b = 0;
166
167     let dx = x2 - x1;
168     let dy = y2 - y1;
169
170     while (b < dy) {
171         while (a < dx) {
172             do Screen.drawPixel(x1 + a, y1 + b);
173
174             let a = a + 1;
175         }
176
177         let a = 0;
178         let b = b + 1;
179     }
180
181     return;
182 }
183
184 /** Draws a filled circle of radius r around (cx, cy). */
185 function void drawCircle(int cx, int cy, int r) {
186     var int dy, sqrt, x1, x2, y;
187
188     if ((cx < 0) | (cx > 511) | (cy < 0) | (cy > 255)) {
189         do Sys.error(12); // Screen.drawCircle: Illegal center coordinates
190     }
191
192     if ((r < 0) | (r > 127) | ((cx - r) < 0) | ((cx + r) > 511) |
193         ((cy - r) < 0) | ((cy + r) > 255)) {
194         do Sys.error(13); // Screen.drawCircle: Illegal radius
195     }
196
197     let dy = -r;
198
199     while (~(dv = r)) {
```

```
200     let sqrt = Math.sqrt((r * r) - (dy * dy));
201
202     let x1 = cx - sqrt;
203     let x2 = cx + sqrt;
204
205     let y = cy + dy;
206
207     do Screen.drawLine(x1, y, x2, y);
208
209     let dy = dy + 1;
210 }
211
212 return;
213 }
214 }
215
```