SVG to PNG

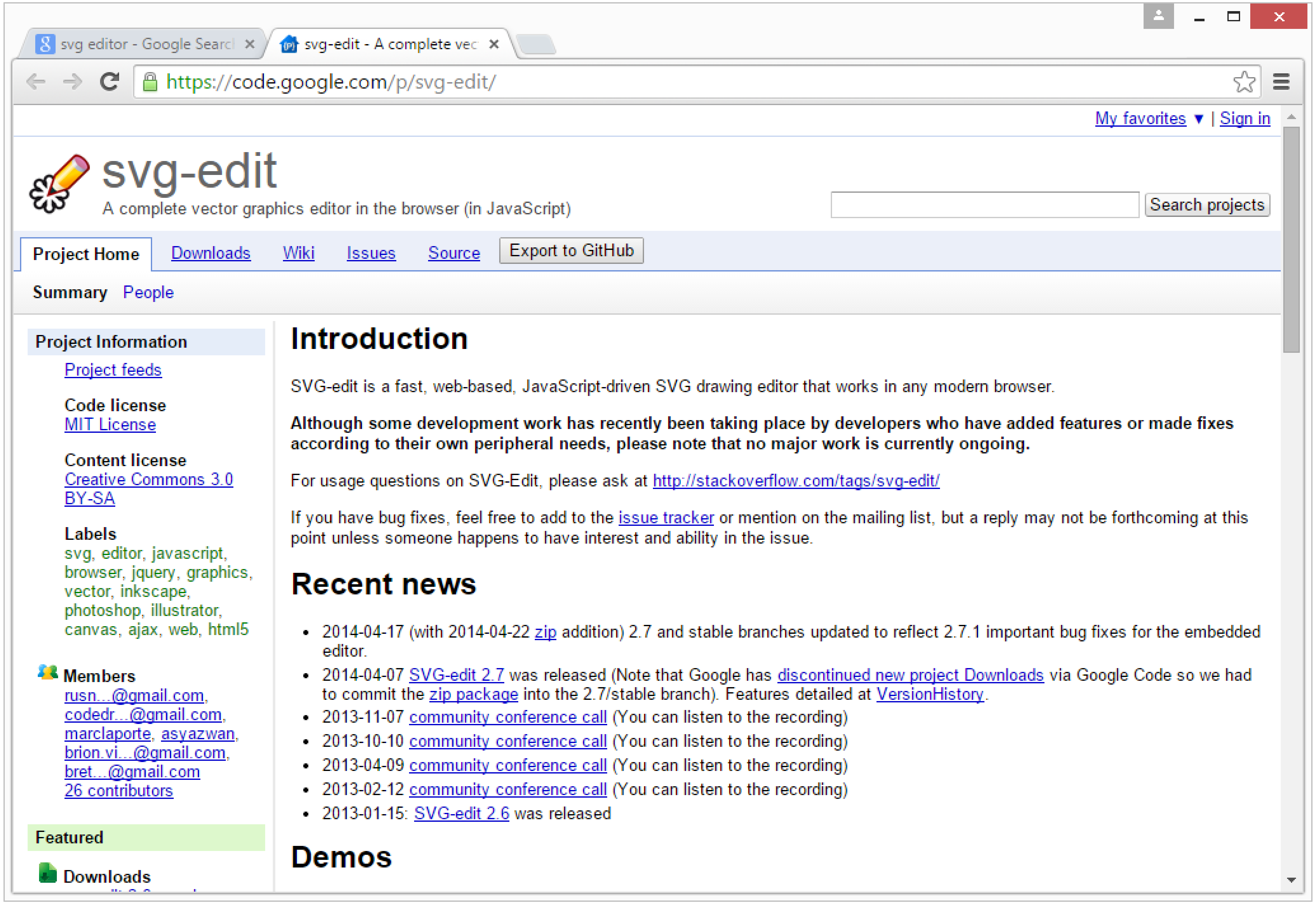
CS6025 Data Encoding

Yizong Cheng

4-2-15

Scalable Vector Graphics

- XML-based vector image format
- Developed by W3C since 1999



svg-edit

A complete vector graphics editor in the browser (in JavaScript)

Project Information

Project feeds

Code license
MIT License

Content license
Creative Commons 3.0 BY-SA

Labels
svg, editor, javascript, browser, jquery, graphics, vector, inkscape, photoshop, illustrator, canvas, ajax, web, html5

Members
rusn...@gmail.com, codedr...@gmail.com, marclaporte, asyazwan, brion.vi...@gmail.com, bret...@gmail.com
26 contributors

Featured

Downloads

Introduction

SVG-edit is a fast, web-based, JavaScript-driven SVG drawing editor that works in any modern browser.

Although some development work has recently been taking place by developers who have added features or made fixes according to their own peripheral needs, please note that no major work is currently ongoing.

For usage questions on SVG-Edit, please ask at http://stackoverflow.com/tags/svg-edit/

If you have bug fixes, feel free to add to the issue tracker or mention on the mailing list, but a reply may not be forthcoming at this point unless someone happens to have interest and ability in the issue.

Recent news

- 2014-04-17 (with 2014-04-22 zip addition) 2.7 and stable branches updated to reflect 2.7.1 important bug fixes for the embedded editor.
- 2014-04-07 SVG-edit 2.7 was released (Note that Google has discontinued new project Downloads via Google Code so we had to commit the zip package into the 2.7/stable branch). Features detailed at VersionHistory.
- 2013-11-07 community conference call (You can listen to the recording)
- 2013-10-10 community conference call (You can listen to the recording)
- 2013-04-09 community conference call (You can listen to the recording)
- 2013-02-12 community conference call (You can listen to the recording)
- 2013-01-15: SVG-edit 2.6 was released

Demos

UNIVERSITY OF Cincinnati

Path Element with Attributes

```
<path d="m 0,0 -1.622,0 0,-6.992 c -1.884,2.569 -3.766,4.812 -5.583,6.982 L -
7.211,0 l -1.504,0 0,-9.911 1.654,0 0,7.037 c 1.775,-2.43 3.604,-4.627
5.375,-6.753 l 0.237,-0.284 1.48,0 L 0.031,0 0,0 z" id="path28521"
style="fill:#000000;fill-opacity:1;fill-rule:nonzero;stroke:none" />
```

C:\classes\602514\Wik
C:\classes\602514\Wikipedi...

```
<svg xmlns="http://www.w3.org/2000/svg" width="296" height="296">
		<path d="M32,236v-28h56v56H32V236L32,236z M80,236v-20H40v40h40V236L80,236z M48,236v-
12h24v24H48V236L48,236z M104,260v-4h-8v-16h8v-24h8v-8H96v-8H64v-8h8v-8H56v16h-8v-8H32v-8h16v-16h-8v8h-8v-
16h16v8h8v8h8v-8h8v8h16v-8h-8v-8H56v-8h24v-8H56v-8h-8v8H32v-24h8v8h48v-8h-8v-8H64v8h-8v-8H40V96h16v16h8v-
8h16v-8h8v8h-8v8h8v8h8v-16h8v-8h-8V72h16v8h8v8h-8v8h8v48h-16v-8h-8v8h-8v8h-8v8h8v8h8v8h16v-8h-8v-8h-8v-
8h16v16h8v-16h8v16h8v-24h8v-8h8v8h-8v8h8v8h24v-8h-8v-8h-8v-16h-8v-8h8v-8h8v-8h-8v-8h-8v16h-8v-8h-8v8h-8v-8h8v-
8h-8V72h-8v-8h8v8h8V40h8v16h16v-8h-8V32h16v8h-8v8h16v-8h8v-8h16v24h-16v-8h-8v16h8v24h8v-8h8v40h16v-8h-
8V96h16v24h16v-8h-8V96h8v16h8v-8h16v16h-8v-8h-8v8h-8v24h8v8h-8v8h-16v8h-8v8h-16v-8h-8v-8h-
8v16h8v8h24v8h16v-8h-8v-8h8v-8h8v8h8v8h8v-16h-8v-8h16v24h-8v16h8v16h-8v24h8v8h-24v16h-24v-8h16v-8h-16v-16h-
8v16h-8v8h8v8h-16v-24h-8v16h-8v-8h-8v-32h8v24h8v-24h8v-16h-8v-8h-8v-8h8v-8h-8v-8h-8v32h8v8h-16v16h-8v16h8v8h-
8v8h16v8h-16v-8h-8v-8h-8v16h-32V260L104,260z M128,248v-8h8v-24h-16v8h8v8h-16v8h-8v8h8v8h16V248L128,248z
M240,240v-8h8v-16h8v-8h-8v-24h-8v24h8v8h-8v8h-8v24h8V240L240,240z M200,236v-4h-8v8h8V236L200,236z
M152,220v-4h-8v8h8V220L152,220z M224,212v-12h-24v24h24V212L224,212z M208,212v-4h8v8h-8V212L208,212z
M144,204v-4h16v-8h-16v-8h-8v8h8v8h-16v-8h-8v8h-8v-8h-8v-8h-8v-8h-8v8h-8v8h8v-8h8v8h8v8h8v8h32V204L144,204z
M120,180v-4h-8v8h8V180L120,180z M160,176v-8h-16v8h8v8h8V176L160,176z M208,164v-4h-8v8h8V164L208,164z
M224,156v-4h8v-24h-8v8h-8v8h-8v-8h-16v-8h-8v-8h8V96h-8v-8h-8v-8h-8v8h-8V64h8v8h8v-8h-8v-8h-8v8h-8v24h8v8h8v-
8h8v24h-8v8h-8v8h8v16h8v-8h16v8h8v8h16v8h8V156L224,156z M216,148v-4h8v8h-8V148L216,148z M88,140v-4h8v-8h-
8v8h-8v8h8V140L88,140z M112,124v-4h-8v8h8V124L112,124z M112,84v-4h-8v8h8V84L112,84z M144,80v-8h-
8v16h8V80L144,80z M192,44v-4h-8v8h8V44L192,44z M256,260v-4h8v8h-8V260L256,260z M256,144v-8h-8v-8h8v8h8v16h-
8V144L256,144z M32,60V32h56v56H32V60L32,60zM80,60V40H40v40h40V60L80,60z M48,60V48h24v24H48V60L48,60z
M208,60V32h56v56h-56V60L208,60z M256,60V40h-40v40h40V60L256,60zM224,60V48h24v24h-24V60L224,60z M96,60v-
4h8v8h-8V60L96,60z M112,52v-4h-8V32h8v8h8v-8h8v8h-8v16h-8V52L112,52z"/></svg>
```

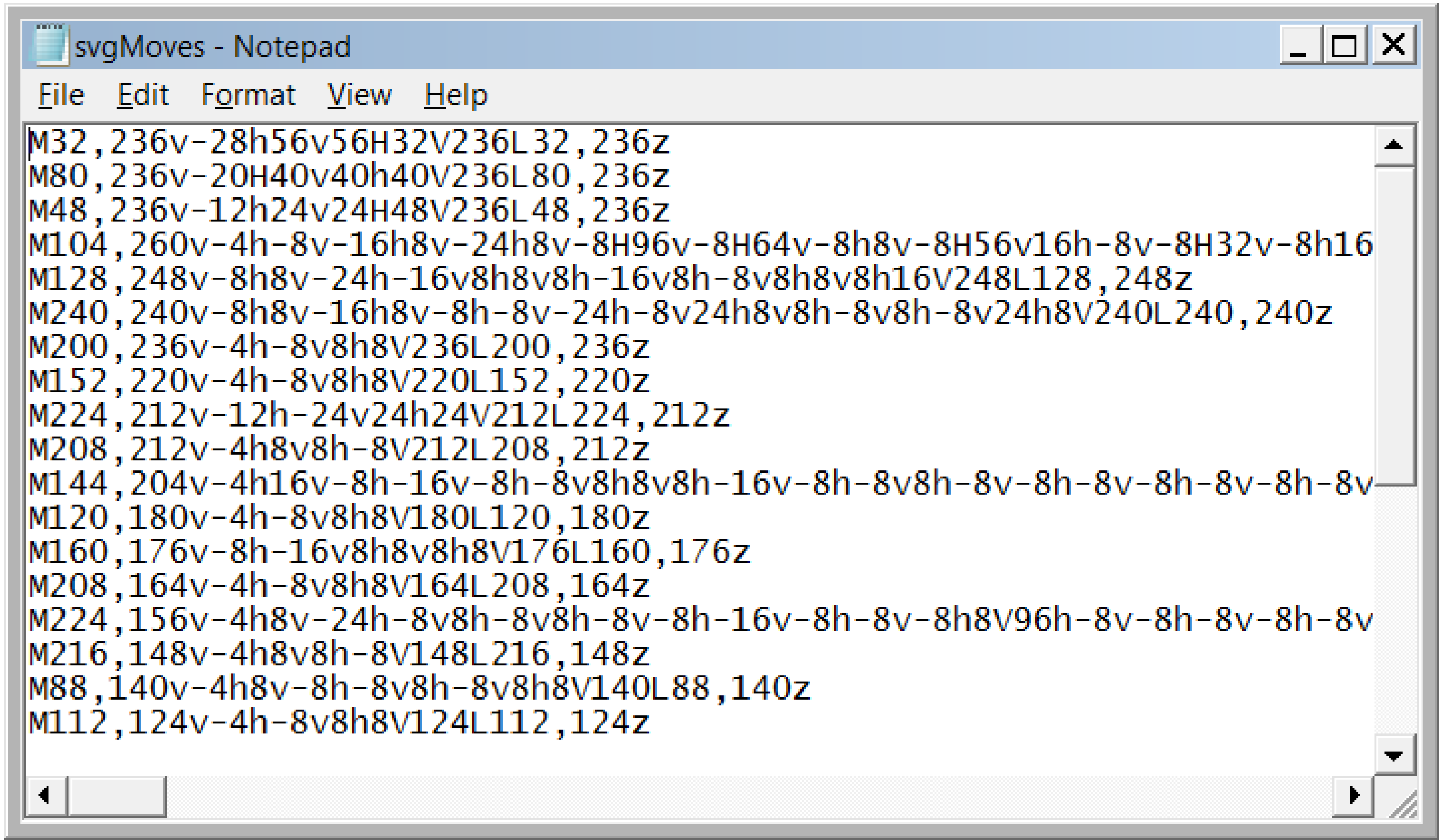
H2O.java

```
public class H2O{

   static final int bufferSize = 4096;
   static final String svgTag = "<svg";
   static final String widthAtt = "width=";
   static final String heightAtt = "height=";
```

```
void readSize(){ // read width and height
     try {
       dataLength = System.in.read(buffer);
     } catch (IOException e){
       System.err.println("IOException");
       System.exit(1);
     }
     int tagClosing = 4;
     while (buffer[tagClosing++] != '>');
     String headTag = new String(buffer, 0, tagClosing);
     if (headTag.indexOf(svgTag) != 0){
       System.err.println(" not a svg file");
       System.exit(1);
     }
     int pos = headTag.indexOf(widthAtt);
     int pos2 = headTag.indexOf('"', pos + 7);
     width = Integer.parseInt(headTag.substring(pos + 7, pos2));
     pos = headTag.indexOf(heightAtt);
     pos2 = headTag.indexOf('"', pos + 8);
     height = Integer.parseInt(
        headTag.substring(pos + 8, pos2));
     dataPosition = tagClosing;
}
```

Polygons in <path >



```
M32,236v-28h56v56H32V236L32,236z
M80,236v-20H40v40h40V236L80,236z
M48,236v-12h24v24H48V236L48,236z
M104,260v-4h-8v-16h8v-24h8v-8H96v-8H64v-8h8v-8H56v16h-8v-8H32v-8h16
M128,248v-8h8v-24h-16v8h8v8h-16v8h-8v8h8v8h16V248L128,248z
M240,240v-8h8v-16h8v-8h-8v-24h-8v24h8v8h-8v8h-8v24h8V240L240,240z
M200,236v-4h-8v8h8V236L200,236z
M152,220v-4h-8v8h8V220L152,220z
M224,212v-12h-24v24h24V212L224,212z
M208,212v-4h8v8h-8V212L208,212z
M144,204v-4h16v-8h-16v-8h-8v8h8v8h-16v-8h-8v8h-8v-8h-8v-8h-8v-8h-8v
M120,180v-4h-8v8h8V180L120,180z
M160,176v-8h-16v8h8v8h8V176L160,176z
M208,164v-4h-8v8h8V164L208,164z
M224,156v-4h8v-24h-8v8h-8v8h-8v-8h-16v-8h-8v-8h8V96h-8v-8h-8v-8h-8v
M216,148v-4h8v8h-8V148L216,148z
M88,140v-4h8v-8h-8v8h-8v8h8V140L88,140z
M112,124v-4h-8v8h8V124L112,124z
```

Moves

- M followed by x,y: move to (x,y)
- L followed by x,y: line from here to (x,y)
- V followed by y: vertical line from here to y
- v followed by dy: move vertically by dy
- H followed by x: horizontal line from here to x
- h followed by dx: move horizontally by dx
- z: line to initial position to complete the polygon

Read Paths

```
String nextPath(){ // from M to z
    int pos = dataPosition; for (; pos < dataLength; pos++)
     if (buffer[pos] == 'M') break;
    if (pos == dataLength) return null;
    int pos2 = pos; for (; pos2 < dataLength; pos2++)
     if (buffer[pos2] == 'z') break;
    dataPosition = pos2 + 1;
    return new String(buffer, pos, pos2 - pos + 1);
  }

  void allPaths(){
    String path = null;
    while ((path = nextPath()) != null)
     process(path);
  }
```

Commands M and L in Path

```
void process(String move){
     int j = 0;
     int command = move.charAt(j);
     int x0 = -1, y0 = -1;
     int x1 = -1, y1 = -1, x2 = -1, y2 = -1;
     while (command != 'z'){
       int i = j + 1;
       for (j++; ; j++) if (move.charAt(j) > '9') break;
       switch (command){
         case 'M': case 'L': int comma = move.indexOf(',', i);
            x2 = Integer.parseInt(move.substring(i, comma));
            y2 = Integer.parseInt(move.substring(comma + 1, j));
            if (command == 'M'){ x0 = x2; y0 = y2; }
            break;
```

Vertical and Horizontal Segments

```
  case 'V':
     y2 = Integer.parseInt(move.substring(i, j));
     x2 = x1;
     break;
  case 'H':
     x2 = Integer.parseInt(move.substring(i, j));
     y2 = y1;
     break;
  case 'v':
     y2 = y1 + Integer.parseInt(move.substring(i, j));
     x2 = x1;
     break;
  case 'h':
     x2 = x1 + Integer.parseInt(move.substring(i, j));
     y2 = y1;
     break;
  default: ;
}
```

Interpretation of Polygon Drawing

- Initialize the image matrix as all white.
- Each time a polygon is read, flip the pixels inside.
- For polygons with vertical and horizontal segments only, equivalent to flip pixels only for horizontal segments between the x values and below the y value.
- Inside gets one flip and outside gets zero or two flips.

Flip Pixels for Each Horizontal Segment

```
          qr = new boolean[height][width]; // initializing
          for (int i = 0; i < height; i++)
           for (int j = 0; j < width; j++) qr[i][j] = true;


           if (command != 'M' && x1 != x2)
              flipPixels(y2, x1, x2);
           x1 = x2; y1 = y2;
           command = move.charAt(j);
           i = j + 1;
         }
         if (x0 != x2) flipPixels(y2, x0, x2);
       }

     void flipPixels(int y, int x1, int x2){
         if (x2 < x1){ int t = x1; x1 = x2; x2 = t; }
         for (int i = x1; i < x2; i++)
          for (int j = 0; j < y; j++) qr[i][j] = !qr[i][j];
     }
```

Print Out the Image Matrix

```
void drawQR(){
   for (int i = 0; i < height / 8; i++){
    for (int j = 0; j < width / 8; j++)
      if (qr[i * 8][j * 8]) System.out.print("X");
      else System.out.print(" ");
    System.out.println();
  }
 }
```

```
XXXXXXX  X X   XX  XX XXXXXXX
X     X  XX  X X  X X X     X
X XXX X   X  X  XX XX X XXX X
X XXX X X    XXX X    X XXX X
X XXX X    X XX X X   X XXX X
X     X XX  X X XXX   X     X
XXXXXXX X X X X X X X XXXXXXX
        XX  XXXX X XX
 XX   X  XX  XX XX  X XX X
 XX XX  XXX X X  X  X XX X XX
   X  X XXX     X   X  X  X X
X      XX X    X   XXXXXXX
XXXXXXX XXX     X   XXXX X X
XX XXX X XX   X X XX  X  X  X
      X      X   XXXX  X XX X
   XXX  XX X XX   XXXXX XX
XX    X  X X XXXXXX X XX   XX
X X X  X  XXXX  X X  XX  X  X
  XXXXX XX XXXX X  X    XXX X
XXX  X X XXXX XXX XXXXX  X X
  X XXXXX X XX  X  XXXXXXX X
        XX    XXXX  X   XX XX
XXXXXXX   XXXXXX    X X XXX X
X     X  XX  X X X XX   XX X
X XXX X  XXX XX  X XXXXXX  X
X XXX X  X   XX  X X X XX XX
X XXX X X   XXXX XX XX XX XXX
X     X XX  X X   X X    X
XXXXXXX  XXXX  XX   XX XXX  X
```

PNG: IHDR, IDAT, IEND

```
static final int PNGHeaderSize = 33;
static final byte[] signature = new byte[]{
 137 - 256, 80, 78, 71, 13, 10, 26, 10,
 0, 0, 0, 13, 73, 72, 68, 82 };
static byte[] iend = new byte[]{
 0, 0, 0, 0, 'I', 'E', 'N', 'D', 0, 0, 0, 0 };
byte[] PNGHeader = new byte[PNGHeaderSize];
byte[] data = null;
byte[] idat = null;
int compressedDataLength = 0;
```

Greyscale with BitDepth (0100)

```
// fill 4 bytes in buffer at offset with a number
 void fillNumber(byte[] buffer, int offset, long number){
   int k = 0; for (; k < 4; k++){
     buffer[offset + 3 - k] = (byte)(number & 0xff);
     number >>= 8;
 }
 }

 void fillPNGHeader(){
   for (int i = 0; i < 16; i++) PNGHeader[i] = signature[i];
   fillNumber(PNGHeader, 16, width);
   fillNumber(PNGHeader, 20, height);
   PNGHeader[24] = 1;  // bit depth
   for (int i = 25; i < 29; i++) PNGHeader[i] = 0;
   crc32.reset();
   crc32.update(PNGHeader, 12, 17);
   fillNumber(PNGHeader, 29, crc32.getValue());
 }
```

Filter Type Byte = 0, 8 pixels/byte

```
void fillIDAT(){
  int lineWidth = width / 8 + 1; // one byte for filter type
  data = new byte[height * lineWidth];
  idat = new byte[height * lineWidth];
  int lineOffset = 0;
  for (int i = 0; i < height; i++){
    data[lineOffset] = 0;
    for (int j = 0; j < width / 8; j++){
      data[lineOffset + j] = 0;
      for (int k = 0; k < 8; k++){
        data[lineOffset + j] <<= 1;
        if (qr[i][j * 8 + k]) data[lineOffset + j] |= 1;
      }
    }
    lineOffset += lineWidth;
  }
```

Data Length, “IDAT”, and Checksum

```
  idat[4] = 'I'; idat[5] = 'D'; idat[6] = 'A'; idat[7] = 'T';
  Deflater compresser = new Deflater();
  compresser.setInput(data);
  compresser.finish();
  // your code for
  // 1. deflate data into idat at position 8.
  // 2. place compressedDataLength at position 0 of idat
  // 3. compute CRC for idat without the length
  // 4. append CRC after compressed data
  // idat = |length|"IDAT"|compressed data|CRC|
}
```

IEND and Writeout

```
void fillIEND(){
  crc32.reset();
  crc32.update(iend, 4, 4);
  fillNumber(iend, 8, crc32.getValue());
}

void writePNG(){
 try {
  System.out.write(PNGHeader);
  System.out.write(idat, 0, compressedDataLength + 12);
  System.out.write(iend);
 } catch (IOException e){
   System.err.println(e.getMessage());
   System.exit(1);
 }
}
```

H2O.main()

```
  public static void main(String[] args){
       H2O h2O = new H2O();
       h2O.readSize();
       h2O.allPaths();
       h2O.fillPNGHeader();
       h2O.fillIDAT();
       h2O.fillIEND();
       h2O.writePNG();
   }
}
```

Homework 20: due 4-8-15

- Complete fillIDAT() of H20.java and run your program on some svg files of QR codes.
- Submit your source code and generated PNG files for QR codes.