Comp sci 129

Final Project

Desing Proposal

Declan Juliano

The problem

I need an easier way to convert images to gcode. For my cnc machine. Preferably a 1 or 2 software system. Instead of 4+.

I am tired of spending hours fiddling around with different software to reach mediocre results.

Research

After looking into this topic deeper. I found although this problem sounds complex. In practice it is a lot of image manipulation, and using ready-made libraries to convert file types.  
SVG’s are just XML files with ready made paths. Which is what Gcode is. A path.

Gcode for those who don’t know.

It is a text file of simple commands, that a machine reads and acts upon.  
Each line of a gcode file, has one of the following commands (there are many more but these are the basics)  
**G**- move like **G0 X10 Y10 Z0 (move to x10 y10 z0)**

**M** – do something like **M106** (enable laser)

**G28** – home tool head.

Stringing these commands together yield a large connect the dots image or 3d structure.  
That large cnc mills or small 3d printers can understand to produce something.

With that out of the way.

The current solution

* Draw image in art program of choice (krita)
* Import the image into photoshop or another vector program
* Perform a trace operation
* Export the vector
* Import the vector into LightBurn or other similar 2d slicer program
* Move and resize
* Export gcode
* Import gcode into machine software.
* Execute gcode

This is standard, and is very fiddly.  
Lots of settings need to changed to get the trace operation in photoshop to comply with the desired gcode export in LightBurn. As well as getting the detail right but not too “blobby”

when we sharpen an SVG trace operation too much, the result is lines made of blobs, instead of being one continuous line.

If we don’t sharpen enough, then the SVG is undetailed and “organic” looking

My justification

The want for a better solution should be quite clear now. There is a better way. And it surprises me there are not many options out there.   
This new software will benefit many who own CNC machines as a business or hobby, by streamlining the process.

And making the experience of using a cnc machine more manageable and less scarry, to ease others into this space.

The New Solution

* Draw image (krita)
* Import image into my program
* Scale and move images
* Export gcode
* Execute gcode to connected machine directly in this program

Two programs are needed for this new solution.

One drawing program

And mine.

Settings will auto populate with the best option in mind, but allow for tweaking for better results if needed.

How?

I have already written most of the program.   
The only bits left are gctrl integration(gcode interpreter) (which I basically rewrote, from the ground up yay!)

And the function that converts the SVG to the Gcode.

This program is written in java on Processing 4 (<https://processing.org/>) for ease of file management and installation on other systems (mac, Linux or windows)

1. //import

2. import g4p\_controls.\*;

3. import jankovicsandras.imagetracer.ImageTracer;

4.

5. //file stuff

6. PrintWriter imagesDropdown;

7. //variables

8. int xScale, yScale;

9. float imgWidth, imgHeight, imgScaler;

10. File file = null;

11.

12. String CurrentImageSelected;

13.

14. //graphics

15. PImage imageSelected;

16.

17. PGraphics PrintArea;

18. PGraphics UI;

19. PGraphics Preview;

20. PGraphics Outline;

21. PGraphics Grid;

22.

23. //variables to deal with unit conversions pixels to mm

24. //change these to fit the x y volume of your machien

25. int bedWidth=250; //mm

26. int bedHeight=150; //mm

27. int gridSize = 20; //pixels

28.

29. //auto change do not change

30. int PPmm; //pixels per mm

31. boolean isProportional = true;

32. float changeAmount;

33.

34. //master list

35. ArrayList<imgData> Images = new ArrayList();

36.

37.

38.

39. //true if the selectImage function is running

40. boolean currentlySelectingImg =false;

41. //string for packaged data

42. //String[] ImageDataString;

43. //stores all data for an image

44. public class imgData {

45. private String address;

46. private int xPos;

47. private int yPos;

48. private int wide;

49. private int tall;

50. private int number;

51. private float scaleFactor;

52. private int scaleWide;

53. private int scaleTall;

54. imgData(String Address, int X, int Y, int Wide, int Tall, int Number) {

55. address = Address;

56. xPos=X;

57. yPos=Y;

58. wide = Wide;

59. scaleWide = Wide;

60. tall = Tall;

61. scaleTall = Tall;

62. number = Number;

63. }

64. String reqData() {

65. return "x pos "+xPos+" y pos "+yPos +"\nwidth " +wide+ " height " +tall+"\naddress " +address + "\nnumber " + number;

66. }

67. String reqAddress() {

68. return address;

69. }

70. Integer reqNum() {

71. return number;

72. }

73. int reqScaleWide() {

74.

75. return scaleWide;

76. }

77. int reqScaleTall() {

78. return scaleTall;

79. }

80. void move(int x, int y) {

81. xPos = x;

82. yPos=y;

83. }

84. void setScale(float amount) {

85. scaleFactor = amount;

86. scaleWide=int(wide/scaleFactor);

87. scaleTall=int(tall/scaleFactor);

88. }

89. }

90. //start

91. void setup() {

92. size(2500, 1300);

93. xScale=width;

94. yScale=height;

95. PPmm=yScale/bedHeight;

96. // surface. setResizable(true);

97. PrintArea = createGraphics(xScale-xScale/4, yScale);

98. UI = createGraphics(xScale/4, yScale);

99. Preview =createGraphics(xScale/4, xScale/4);

100. Outline = createGraphics(xScale-xScale/4, yScale);

101. Grid = createGraphics(xScale-xScale/4,yScale);

102. createGUI();

103.

104. //initializeDropdown();

105. drawImages();

106. println("current dimensions are: " + yScale+ "pixels tall to" + bedHeight + "mm bed height, making a PPmm of" + PPmm);

107. }

108.

109. //loop

110. void draw() {

111. //if the window changes size resize

112. if (xScale!=width||yScale!=height) {

113. xScale=width;

114. yScale=height;

115. println(width+" "+height);

116. UI();

117. }

118. //redraw every frame

119. cleanUI();

120. image(Grid,xScale/4,0);

121. image(PrintArea, xScale/4, 0);

122. image(Outline, xScale/4, 0);

123.

124.

125. if (currentlySelectingImg) {

126. drawImages();

127. }

128. }

129.

130. void mousePressed() {

131. }

132.

133.

134. void keyPressed() {

135.

136. if (key=='f'||key =='F') selectInput("select Image:", "ImageSelected", file);

137. if (key =='r'||key =='R')drawImages();

138. if (key == 'd'||key == 'D') packdata();

139. if (key == 'p'||key =='P')createPrevImg(ImageDropdown.getSelectedText());

140. if(key=='e'||key=='E') selectOutput("slected where to save the png to be converted","export",file);

141. if(key=='t'||key=='T')selectInput("select Image to be traced","trace",file);

142. }

143.

144. void printArea() {

145. }

146.

147. //executes when image is selected

148. void ImageSelected(File file) {

149. try {

150. imageSelected = loadImage(file.getAbsolutePath());

151. //add new image to master list

152. Images.add(new imgData(file.getAbsolutePath(), 0, 0, imageSelected.width, imageSelected.height, Images.size()));

153. println("begin image data");

154. println(Images.get(Images.size()-1).reqData());

155. println("end image data");

156. println(Images.size());

157. //enable the selected image so we can draw all of the images

158. currentlySelectingImg=true;

159. //update the sliders to the new image position

160. updateSliders(ImageDropdown.getSelectedIndex());

161. //sets the preview as the new image selected.

162. createPrevImg(file.getAbsolutePath());

163. }

164. catch(Exception e) {

165. //throw error if not an image

166. System.err.println("oops woopsy, make sure its an image. you dumb pone");

167. }

168. }

169.

170. //update images

171. void drawImages() {

172. //draw the grid

173. drawGrid();

174. PrintArea.beginDraw();

175. PrintArea.clear();

176. //txt file for dropdown

177. imagesDropdown = createWriter("data/images.txt");

178.

179. //disp images

180. int i=0;

181. for (imgData img : Images) {

182. i++;

183.

184. println("currently processing image");

185. println(img.address);

186.

187. //writes img.adress to dropdown

188. imagesDropdown.println(img.address);

189. imageSelected = loadImage(img.address);

190. PrintArea.image(imageSelected, img.xPos, img.yPos, img.scaleWide, img.scaleTall);

191. }

192. println("total images processed: "+i);

193.

194. PrintArea.endDraw();

195. imagesDropdown.flush();

196. //display the print area with the new images

197. image(PrintArea, xScale/4, 0);

198. println("draw fin");

199. //update the dropdown

200. ImageDropdown.setItems(loadStrings("data/images.txt"), ImageDropdown.getSelectedIndex());

201. cleanUI();

202.

203. //remove the placement outline

204. Outline.beginDraw();

205. Outline.clear();

206. Outline.endDraw();

207.

208. //disable imageselecting to stop running drawImages()

209. currentlySelectingImg = false;

210.

211. }

212.

213. //pack data to a new array list so we can export for saving the season

214. void packdata() {

215. ArrayList ImageDataString = new ArrayList<String>();

216. for (imgData img : Images) {

217. ImageDataString.add(img.reqData());

218. }

219. println(ImageDataString.toString());

220. // saveStrings("Images.txt", ImageDataString);

221. }

222.

223.

224. //\*\*\*\*\*\*\*UI\*\*\*\*\*\*\*

225. //initializes and resizes the ui strip

226. void UI() {

227. PrintArea = createGraphics(xScale-xScale/4, yScale);

228. UI = createGraphics(xScale/4, yScale);

229. Preview = createGraphics(xScale/4, xScale);

230. UI.beginDraw();

231. UI.background(0);

232. UI.endDraw();

233.

234.

235. image(UI, 0, 0);

236. //clears garbage memory

237. System.gc();

238.

239. drawImages();

240. }

241. //cleans the ui strip

242. void cleanUI() {

243. UI.beginDraw();

244. UI.background(100);

245. UI.endDraw();

246. image(UI, 0, 0);

247. image(Preview, 0, yScale-xScale/4);

248. }

249.

250. //draw the grid

251. void drawGrid(){

252. Grid.beginDraw();

253. Grid.background(255);

254. //txt file for dropdown

255.

256. //grid

257. for (int gridX=0; gridX<width; gridX+= gridSize) {

258. Grid.stroke(200, 200, 255);

259. Grid.line(gridX, 0, gridX, height);

260. }

261. for (int gridY=0; gridY< height; gridY+= gridSize) {

262. Grid.stroke(200, 200, 255);

263. Grid.line(0, gridY, width, gridY);

264. }

265.

266. Grid.endDraw();

267. }

268.

269. //draws the image preview givent the number assosiated with adress assosiated with the image

270. void createPrevImg(String address) {

271.

272. imageSelected = loadImage(address);

273.

274. Preview.beginDraw();

275. Preview.background(135);

276.

277. //resize the image to fit the area

278. println("img width: "+imageSelected.width +"img height: "+imageSelected.height);

279. if (imageSelected.width>=imageSelected.height) {

280. println("wide");

281. imgScaler=(float)imageSelected.width/Preview.width;

282. imgWidth=(float)imageSelected.width/imgScaler;

283. imgHeight=(float)imageSelected.height/imgScaler;

284. } else if (imageSelected.width<imageSelected.height) {

285. println("tall");

286. imgScaler=(float)imageSelected.height/Preview.height;

287. imgWidth=(float)imageSelected.width/imgScaler;

288. imgHeight=(float)imageSelected.height/imgScaler;

289. }

290. Preview.image(imageSelected, 0, 0, imgWidth, imgHeight);

291. println("scaler: " + imgScaler);

292. println("new img width: "+imgWidth +"new img height: "+imgHeight);

293. println("preview width: "+Preview.width + "preveiw height: "+Preview.height);

294. println("Preview image");

295. println(address);

296. Preview.endDraw();

297. image(Preview, 0, yScale-xScale/4);

298. }

299.

300.

301. //move the image on the xy plane

302. void moveImage(int index, float xPercent, float yPercent) {

303. imgData img = Images.get(index);

304.

305. int Xpos = int(map(xPercent, 0, 1, 0, PrintArea.width));

306. int Ypos = int(map(yPercent, 0, 1, 0, PrintArea.height));

307. drawOutline(img, Xpos, Ypos);

308. img.xPos=Xpos;

309. img.yPos=Ypos;

310. //displpay the outline

311. image(Outline, xScale/4, 0);

312. }

313.

314. //scale image horizontaly

315. void scaleImageX(int index, float xAmount) {

316. imgData img = Images.get(index);

317. changeAmount =(float)img.wide/(xAmount\*PPmm);

318. //if proportional check box is checked. then scale uniformly

319. if (isProportional) {

320. img.setScale(changeAmount);

321. } else {

322. //just scale width

323. img.scaleWide = int(img.wide/changeAmount);

324. }

325. //draw the outline

326. drawOutline(img, img.xPos, img.yPos);

327. }

328.

329. //scale image vertiacaly

330. void scaleImageY(int index, float yAmount) {

331. imgData img = Images.get(index);

332. changeAmount =(float)img.tall/(yAmount\*PPmm);

333. //if proportional editing then scale uniformly

334. if (isProportional) {

335. img.setScale(changeAmount);

336. } else {

337. //just scale height

338. img.scaleTall = int(img.tall/changeAmount);

339. }

340. //draw the outline

341. drawOutline(img, img.xPos, img.yPos);

342. }

343.

344.

345.

346. //resets the data feilds to the current selected image

347. void updateSliders(int index) {

348. imgData img = Images.get(index);

349. xPosSlider.setValue(map(img.xPos, 0, PrintArea.width, 0, 1));

350. yPosSlider.setValue(map(img.yPos, 0, PrintArea.height, 0, 1));

351. WidthBox.setText(img.scaleWide/PPmm+"");

352. HeightBox.setText(img.scaleTall/PPmm+"");

353.

354. //yPosSlider.setValue(map(img.scaleFactor,1,100,0,1));

355. drawOutline(img, img.xPos, img.yPos);

356. print("current" +index+"is being updated");

357. }

358.

359. //function for drawing outline

360. void drawOutline(imgData img, int Xpos, int Ypos) {

361. Outline.beginDraw();

362. Outline.clear();

363. Outline.noFill();

364. Outline.stroke(100, 50, 200);

365. Outline.strokeWeight(10);

366. Outline.rect(Xpos, Ypos, img.scaleWide, img.scaleTall);

367. Outline.endDraw();

368. }

369. //exports the PrintArea layer as a png with transparency

370. void export(File file){

371. println("exporting file " ,file);

372. PrintArea.save(file.getAbsolutePath());

373. trace(file);

374.

375. }

376.

377.

378. //converts the png to the SVG with a .svg extension

379. void trace(File file){

380. try{

381. println("traceing ", file.getAbsolutePath());

382. // print(path.toAbsolutePath());

383. ImageTracer.saveString(file.getAbsolutePath()+".svg",ImageTracer.imageToSVG(file.getAbsolutePath(),null,null));

384. println("Trace success");

385. }catch(Exception e){

386. println(e);

387. }

388. }

389. //this is to convert svg to gcode

390. void svgToGcode( File file){

391. /\*

392. code that will convert the paths found in the svg to gcode.

393. it will convert the path to individual G0 and G1 vectorss.

394. by taking the difference of the origin 0,0 (top left corner of the image) and the current point in the path of the SVG in question.

395. then divide that by the PPmm(pixels per mm). to convert PIxels to mm,

396. then format a string like

397. GO X~~ Y~~

398. \*\*\*\*\*turning on and off the laser\*\*\*\*\*

399. when a new <path> stars we will output a m106 s200. (turn laser on)

400.

401. when the <path> ends aka a new one begins we will output a m106 s0 (turn laser off); then repeat this loop.

402. \*/

403. }

404.

**Very messy just a prototype threw together in a day.**

How the user will interact

This is still very beta,

The plan is to eventually implement click and drag features for image manipulation.

As of now its is a set of sliders, text boxes and buttons.

With keyboard inputs if you don’t want to use the buttons

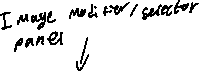
ie:

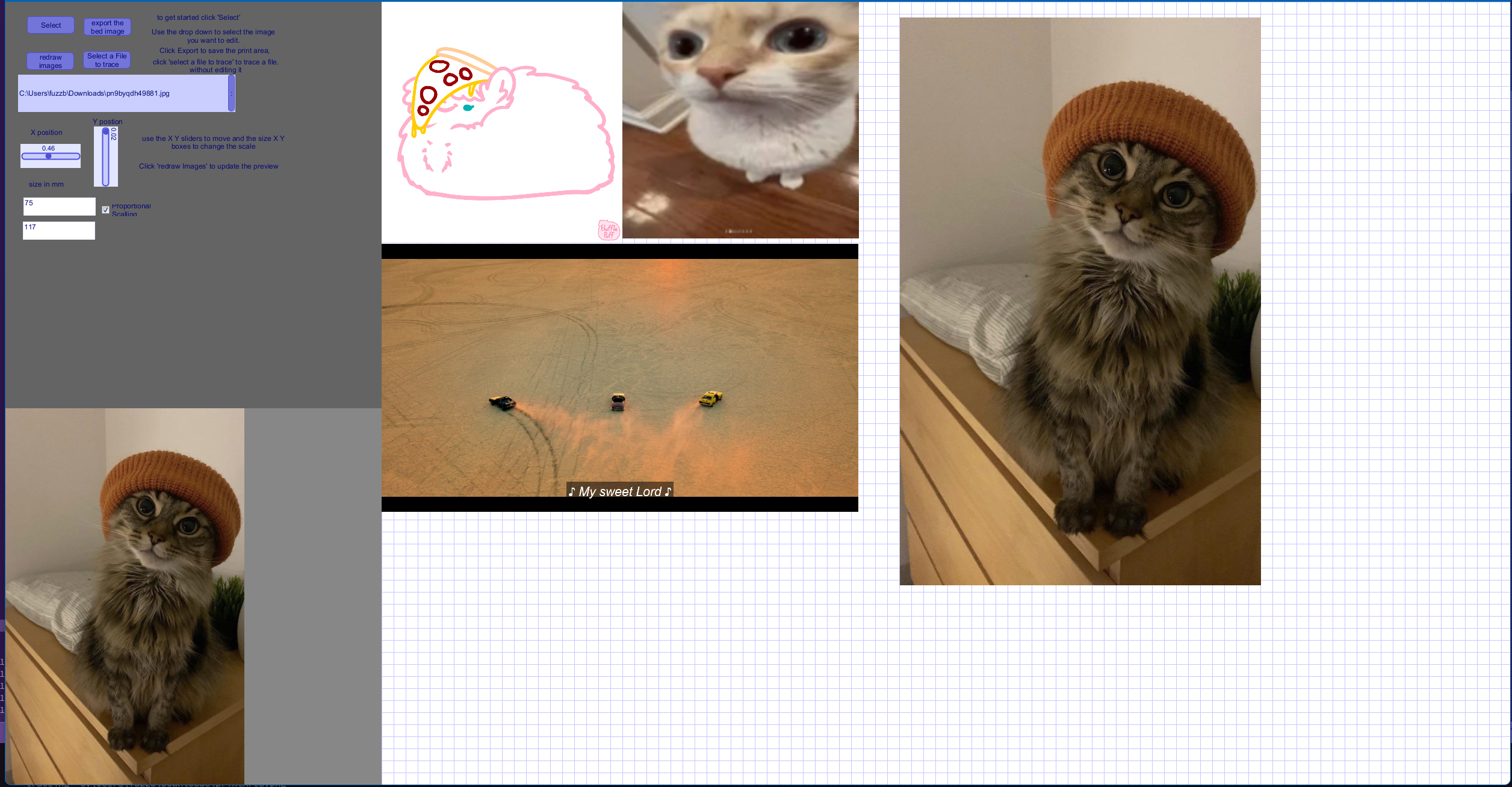
1. if(Key == ‘f’) loadImage();

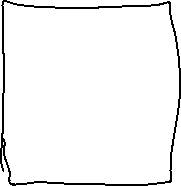
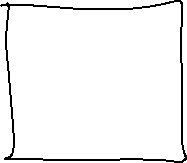
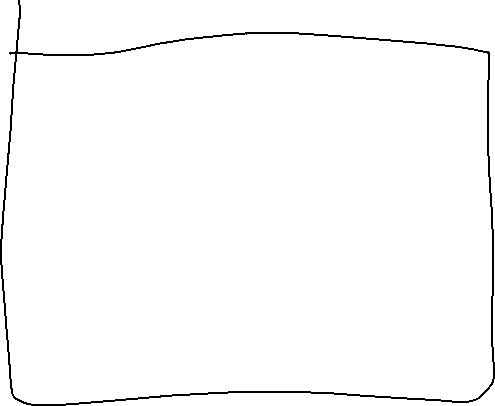
does the same as clicking “select”

Here is the layout with random images.

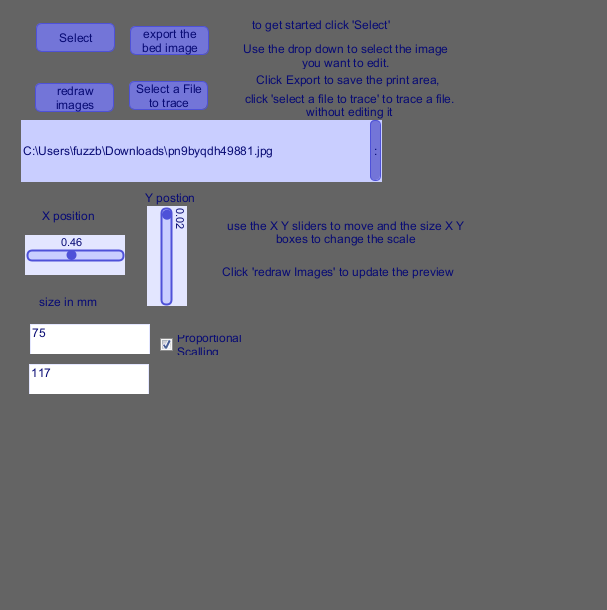
With the different parts outlined







Here is the image selector/modifier panel thing

It has the basics, a load button(select) update button (redraw) and export button and a button to convert any image to svg without needed to import it

Here is the exported SVG of the PrintArea above



This is a very rough prototype of this software as a proof of concept.  
It has worked and shows potential.

Thank you!  
Rest of code can be found on my GitHub at <https://github.com/fuzzbuzzbay999/Gcode-slicer/tree/main>