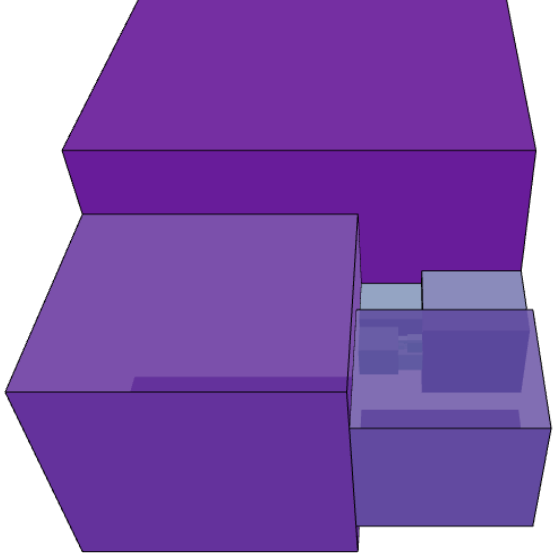

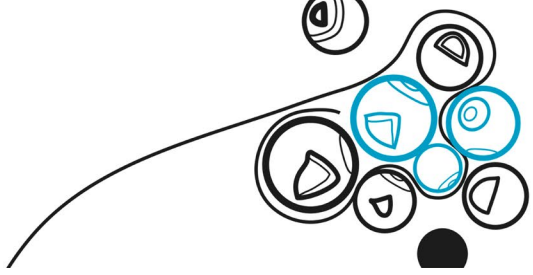

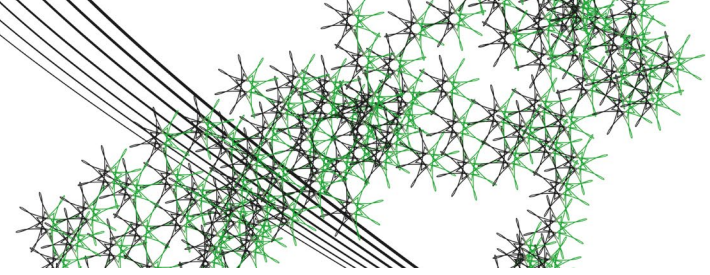

CODE REVIEW MATTER OF PERSPECTIVE

How does one convert an idea into mathematics and form? Unfortionately, there are no 4D printers available (yet?). A printed 3D projection would still be very hard to interpret. As such a rotating 3D projection on a 2D screen was chosen as the best way of displaying the art piece. This poster explains the steps from Fibonacci sequence to screen, to mind.

PROJECTION, YOUR HONOR

When displaying 3D items on a 2D display, the effective trainsform from (x,y,z) to x′, y′ is simply put (ignoring the fancy camera stuff in the background):

$$x' = \frac{X}{z}, \qquad y' = \frac{Y}{z}$$

This same principle holds from 4D to 3D:

$$x' = \frac{X}{\omega}, \qquad y' = \frac{Y}{\omega}, \qquad z' = \frac{Z}{\omega}$$

This principle is leveraged by nking (https://www.openprocessing.org/sketch/205544/) to create an open source sketch which renders and rotates a 4D Hypercube. This code has been modified extensively to render and rotate the output of “nautilus shell” generating code, and has been modified esthetically to create a piece of art.

Spiral Computation

```
// Inspiration:
//https://www.maa.org/sites/default/files/images/upload_library/23/picado/seashells/introdeng.html

int xCords[]= new int[15];
int yCords[]= new int[15];
int fibseq[] = {1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987, 1597, 2584};

void draw() {
  translate(512, 512, 0);
  background(255);
  strokeWeight(1);
  int direction = 1;
  int x = 512;
  int y = 512;
  noFill();
  smooth();
  beginShape();
// Trace the steps in a spiral in steps from the seuqnce
  for (int i = 0; i< 15; i++) {
    switch(direction) {
    case 1:
      x += fibseq[i];
      y += fibseq[i];

      vertex(x, y, 0);
      direction++;
      break;
    case 2:
      x -= fibseq[i];
      y += fibseq[i];

      vertex(x, y, 0);
      direction++;
      break;
    case 3:
      x -= fibseq[i];
      y -= fibseq[i];

      vertex(x, y, 0);
      direction++;
      break;
    case 4:
      x += fibseq[i];
      y -= fibseq[i];

      vertex(x, y, 0);
      direction = 1;
      break;
    }
    println(x, y);
    xCords[i]= x-512;
    yCords[i] =y-512;
  }
  endShape();

  println("X coordinates");
  println(xCords);
  println("Y coordinates");
  println(yCords);
}
```

Translate in 4D

```
void generateCords() {

    int[] XCords = {1, -1, -4, 1, 9, -4, -25};
    int[] YCords = {1, 3, 0, -5, 3, 16, -5};
    boolean isRotated =true;
    int[][][] cords = new int[XCords.length][8][3];

    for (int i =0; i<XCords.length; i++) {
      if (isRotated == false) {
        isRotated = false;

        int[][] cord = {
          {XCords[i], YCords[i]+1, 1, 1},
          {XCords[i], YCords[i]-1, 1, 1},
          {XCords[i], YCords[i]+1, -1, 1},
          {XCords[i], YCords[i]-1, -1, 1},

          {XCords[i], YCords[i]+1, 1, -1},
          {XCords[i], YCords[i]-1, 1, -1},
          {XCords[i], YCords[i]+1, -1, -1},
          {XCords[i], YCords[i]-1, -1, -1}
        };
        cords[i] = cord;
      } else {
        isRotated = true;

        int[][] cord = {
          {XCords[i]+1, YCords[i], 1, 1},
          {XCords[i]-1, YCords[i], 1, 1},
          {XCords[i]+1, YCords[i], -1, 1},
          {XCords[i]-1, YCords[i], -1, 1},

          {XCords[i]+1, YCords[i], 1, -1},
          {XCords[i]-1, YCords[i], 1, -1},
          {XCords[i]+1, YCords[i], -1, -1},
          {XCords[i]-1, YCords[i], -1, -1}
        };
        cords[i] = cord;
      }
    }
```

Poke around in the code yourself?

https://gitlab.com/JelleHamoen/4dfibonacci

Feel free to adapt!

Jelle Hamoen (s2189488) – Art, Math and Technology – 19-06-2019

UNIVERSITY OF TWENTE.