

# INFRA

## Laurent

Trucheteries : pour faire des trucheteries au plotter ou à la brodeuse, il faut joindre des chemins disjoints avec vptype :

```
vptype read truchet_094.svg linemerge linesimplify -t 0.05 write truchet_94_merge.svg
```

## Infrabuble

<https://dev.laurent-malys.fr/bacasable/infra/>

## Multitude

<https://multitude.labomedia.org/>

<https://multitude.exsitu.xyz/v/map/19255da86d5b11f61>

## Dessins génératifs

Broderie + upcycling : [https://www.instagram.com/p/C58HraKiuYE/?img\\_index=1](https://www.instagram.com/p/C58HraKiuYE/?img_index=1)

Trucs à broder: <https://dev.laurent-malys.fr/harmono-bro/>

## Infra PickUp



Profiter de ces beaux jours d'automne pour se chauffer au néons et ressortir la petite [bobineuse](#) pour créer un mirco

## la base infrastructurelle

- Deux disques de cp plaqué de 3 mm découper à la scie cloche
- Un aimant Alnico de 10mm de diamètre
- Du fil enamel de 0,063mm de diamètre
- Une plaque rectangulaire pour insérer les œillets et fixer le câbles audio

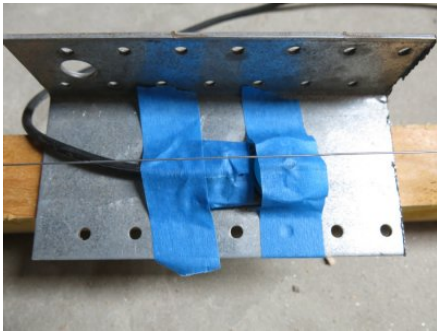
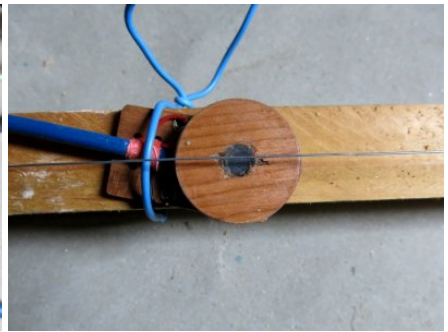
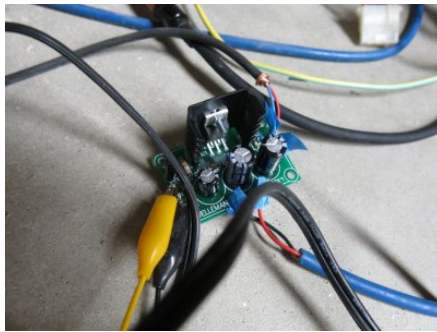
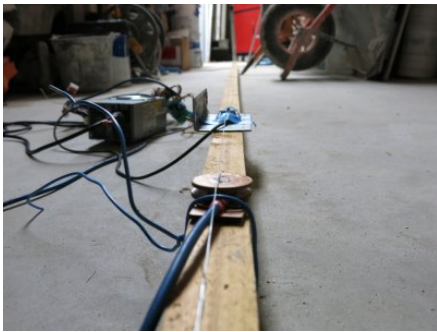
## le bobinage

- Un petit programme arduino pour compter les tour de bobines (grâce à un aimant placer sur l'axe de rotation et interrupteur reed)
  - récupérer ici : <https://forum.arduino.cc/t/simple-pulse-counter/519930>
- Environs 3500 tours





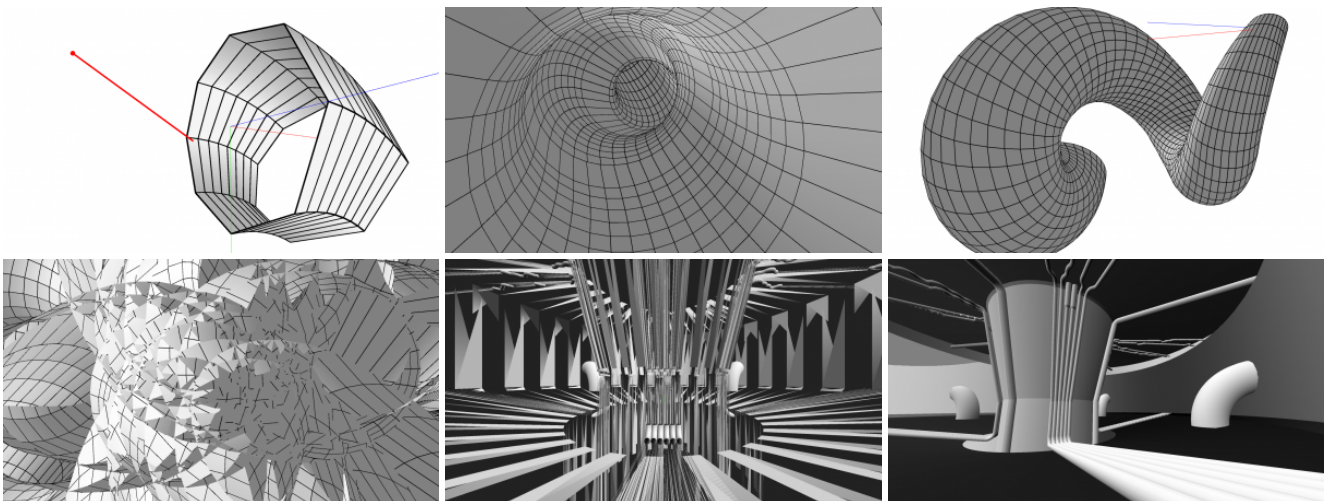
- Ça fonctionne plutôt bien au niveau des infras
- Ajout d'un second micro (caché sous le scotch bleu)
- Câblage du gros micro et du petit avec un ampli mono 7W pour créer un feedback de résonances



## Infra graphique

Dérive dans les images semi-graphiques. Avant le graphisme «hi-res». Journal : [infra-graphique](#)

## Quaternion



## iteration1.pde

```
import queasycam.*;
import com.badlogic.gdx.math.*;
import nervoussystem.obj.*;

QueasyCam cam;

static final Vector3 xAxis = new Vector3(1.0, 0.0, 0.0);
float rootSize = 8;
int numSeg = 128;

Segment root = new Segment(null, 10, new Quaternion());

public void setup() {
    fullscreen(P3D);

    cam = new QueasyCam(this, 1, 9999);
    cam.key_forward = 'o';
    cam.key_left = 'k';
    cam.key_backward = 'l';
    cam.key_right = 'm';
    cam.key_up = 'i';
    cam.key_down = 'p';

    sphereDetail(16);

    Segment c1 = root.branch(rootSize, new Quaternion().setEulerAngles(0, 0, -100));
    Segment c2 = root.branch(rootSize, new Quaternion().setEulerAngles(120, 0, -100));
    Segment c3 = root.branch(rootSize, new Quaternion().setEulerAngles(240, 0, -100));
    for (int i=0; i<numSeg; i++) {
        c1 = c1.branch(rootSize * pow(1.0-i/(20.0*numSeg), i), new Quaternion());
        c2 = c2.branch(rootSize * pow(1.0-i/(20.0*numSeg), i), new Quaternion());
        c3 = c3.branch(rootSize * pow(1.0-i/(20.0*numSeg), i), new Quaternion());
    }

    root.update();
}

public void draw() {
    checkKeys();

    background(255);
    lights();

    noStroke();
    translate(width/2, height/2);

    sphere(50);

    root.draw();
}

void checkKeys() {
    if (keyPressed == false)
        return;

    float step = 0.1;
    Quaternion rotation = new Quaternion();
    if (key == 'a') {
        rotation.setEulerAngles(-step, 0.0, 0.0);
    }
    if (key == 'z') {
        rotation.setEulerAngles(step, 0.0, 0.0);
    }
    if (key == 'q') {
        rotation.setEulerAngles(0.0, -step, 0.0);
    }
    if (key == 's') {
        rotation.setEulerAngles(0.0, step, 0.0);
    }
    if (key == 'w') {
        rotation.setEulerAngles(0.0, 0.0, -step);
    }
    if (key == 'x') {
        rotation.setEulerAngles(0.0, 0.0, step);
    }

    for (Segment seg : root.children) {
        while (seg.hasChildren()) {
            seg.localRot.mul(rotation).nor();
            seg = seg.children.get(0);
        }
        seg.localRot.mul(rotation).nor();
    }

    root.update();
}
```

```

/*****
**      class Segment      **
*****/

public class Segment {
    static final int faces = 32;
    static final float l2b = 10.0; // Length to base ratio

    protected Segment parent = null;
    protected Vector3 head = new Vector3();
    protected Vector3 rootWorldPos = null;

    private Quaternion localRot = new Quaternion(); // Rotation from parent Segment
    private Quaternion globalRot = new Quaternion(); // Total rotation in world space
    private float len;
    private float baseRadius;
    private Vector3[] points = new Vector3[faces];
    private Vector3 tmpVec = new Vector3();

    private ArrayList<Segment> children = new ArrayList();

    public Segment(Segment parent, float len, Quaternion rot) {
        this.parent = parent;
        if (parent == null) {
            rootWorldPos = new Vector3();
            globalRot = rot.cpy();
        } else {
            rootWorldPos = parent.rootWorldPos;
        }
        this.localRot = rot.cpy();
        this.len = len;
        baseRadius = len * l2b * 0.5;
    }

    public Segment branch(float len, Quaternion rot) {
        Segment child = new Segment(this, len, rot);
        children.add(child);
        return child;
    }

    public boolean hasChildren() {
        return !children.isEmpty();
    }

    public void draw() {
        // Draw head line
        stroke(0);
        Vector3 up = new Vector3(0, len*0.4, 0);
        globalRot.transform(up);
        if (parent != null)
            line(parent.head.x, parent.head.y, parent.head.z, head.x, head.y, head.z);
        else
            line(0.0, 0.0, 0.0, head.x, head.y, head.z);

        // Draw UP line
        stroke(255, 0, 0);
        line(head.x, head.y, head.z, head.x + up.x, head.y + up.y, head.z + up.z);

        noStroke();
        if (hasChildren())
            drawSegment();
        else
            drawTip();

        for (Segment child : children)
            child.draw();
    }

    private void drawSegment() {
        beginShape(QUAD_STRIP);
        for (int i=0; i<faces+1; i++) {
            if (parent == null) {
                Quaternion rot = new Quaternion();
                rot.setFromAxisRad(xAxis, i * TWO_PI / faces);
                rot.mulLeft(globalRot);
                rot.transform(tmpVec);
                tmpVec.set(len, baseRadius, 0.0);
            } else {
                tmpVec.set(parent.points[i%faces]);
            }
            vertex(tmpVec.x, tmpVec.y, tmpVec.z);
            tmpVec.set(points[i%faces]);
            vertex(tmpVec.x, tmpVec.y, tmpVec.z);
        }
        endShape();
    }

    private void drawTip() {
        // Draw a capped tip if this segment has no children
        beginShape(TRIANGLE_FAN);
        vertex(head.x, head.y, head.z);
        for (int i=0; i<faces+1; i++) {

```

```

    if (parent == null) {
        Quaternion rot = new Quaternion();
        rot.setFromAxisRad(xAxis, i * TWO_PI / faces);
        rot.mulLeft(globalRot);
        rot.transform(tmpVec);
        tmpVec.set(len, baseRadius, 0.0);
    } else {
        tmpVec.set(parent.points[i%faces]);
    }
}
endShape();
}

public void update() {
    Vector3 pos = rootWorldPos.cpy();
    globalRot.set(localRot);
    if (parent != null) {
        pos.add(parent.head);
        globalRot.mulLeft(parent.globalRot).nor();
    }
    head.set(len, 0.0, 0.0); // Going right by default
    globalRot.transform(head);
    head.add(pos);

    float angle = 0.0;
    Quaternion rot = new Quaternion();
    for (int i=0; i<faces; i++) {
        rot.setFromAxisRad(xAxis, angle);
        rot.mulLeft(globalRot);
        tmpVec.set(len, baseRadius, 0.0);
        rot.transform(tmpVec);
        points[i] = tmpVec.cpy().add(pos);
        angle += TWO_PI / faces;
    }

    for (Segment child : children)
        child.update();
}
}

```

Article extrait de : <http://www.lesporteslogiques.net/wiki/> - **WIKI Les Portes Logiques**

Adresse : [http://www.lesporteslogiques.net/wiki/recherche/residence\\_infra/start?rev=1731839842](http://www.lesporteslogiques.net/wiki/recherche/residence_infra/start?rev=1731839842)

Article mis à jour : **2024/11/17 11:37**