# Polygones dégénérés

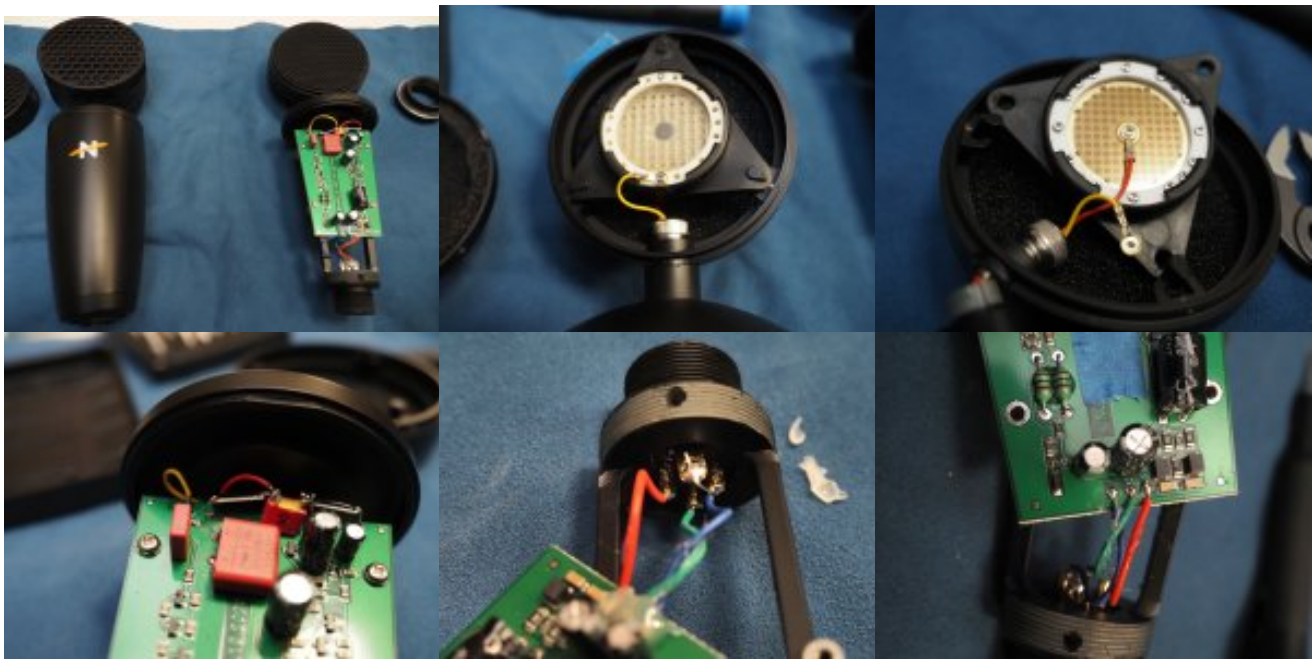
Au fablab @ Flux, 8-12 novembre 2025

Pad : https://annuel.framapad.org/p/rda_polygone_2025

## XY LDC

Alléger une paire de micros larges diaphragmes NEAT King Bee II (6dB Self Noise) de 1kg pièce pour en faire quelque chose de portable pour du field recording.

### NEAT King Bee II en image



XY LDC

## mesh 2 svg 2 paper

mesh2svg2paper (notes)

papercraft (notes)

Utiliser l'add-on Blender «Export Paper Model»

### Petite parenthèse VRML

Visualiseur en ligne pour fichiers VRML : https://imagetostl.com/view-wrl-online#convert
Navigateur VRML à compiler : https://freewrl.sourceforge.io/examples.html
Peut être possible d'afficher un fichier VRML avevc : https://castle-engine.io/
https://www.qiew.org/
https://web.archive.org/web/20140412054654/http://cic.nist.gov/vrml/vbdetect.html

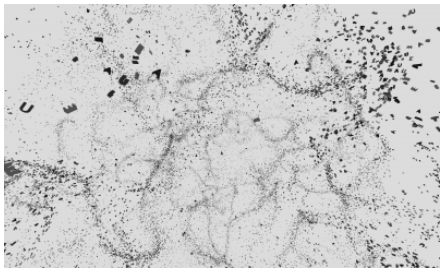
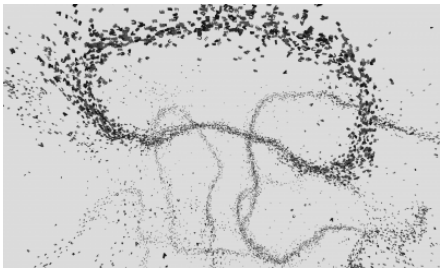
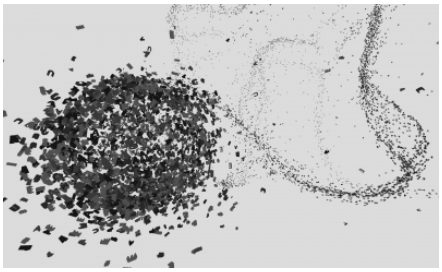
# Physarum Polycephalum

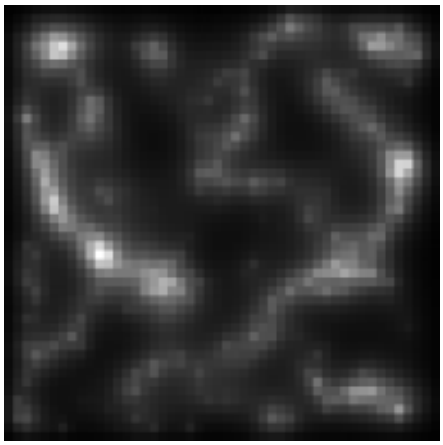
**Maintenant en 3D ! **

Reprise d'un vieux projet de génération de motifs d'après l'algorithme de croissance du Physarum Polycephalum (a.k.a “le blob”), mais en y ajoutant une troisième dimension pour passer du pixel au voxel.

- Godot 4.5
- Utilisation d'un compute shader pour accelerer le calcul de diffusion des traces chimiques.
- Grille 3D relativement petite (32x32x32 à 100x100x100) avec un nombre d'agent/particules allant de 5.000 à 20.000

(code source à venir)



(vue de coupe au centre de la grille de voxels)

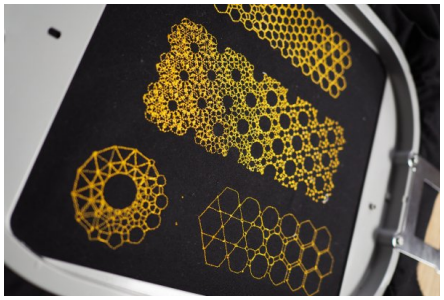
### Travaux similaires (et plus aboutis)

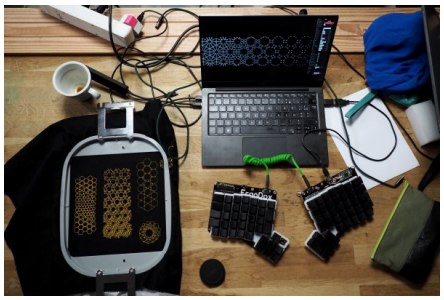
- PolyPhy
- PhysOM

# Pavages de polygones brodés

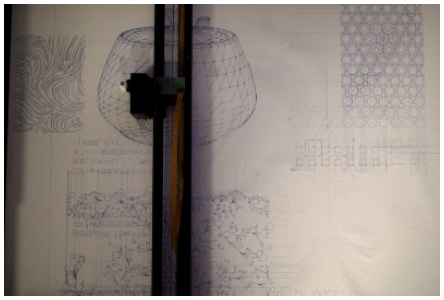
Les fichiers sont sur la clés usb de la machine, dossier 'laurent'







## Roland DPX-3300





### Dessiner un fichier svg depuis linux

#### Conversion de fichiers svg en RP-GL2 avec vpype-rpgl

```
pipx install vpype
pipx inject vpype https://gitlab.com/losylam/vpype-rpgl/-/archive/v0.1.0/vpype-rpgl-v0.1.0.tar.gz
```

Usage

```
vpype read input.svg rpwrite output.rpgl
```

### Streamer le fichier avec hpplot

Installation hpplot

```
git clone https://github.com/rhalkyard/hpplot
cd hpplot
pipx install .
```

Usage

```
hpplot -f query -B 32 /dev/ttyUSB0 output.rpgl
```

### Les trucs qui ne marchent pas

- Le dessin s'arrête avant la fin, il manque les quelques derniers traits et souvent le stylo reste baissé.
- Le dessin est retourné en y (le haut est en bas) ⇒ à corriger dans vpype-rpgl
- Le chargement / changement de stylo ne fonctionne pas
- Il faut placer la feuille en bas à gauche (x=11mm, y=8mm), l'origine est fixée en dur dans le plugin (voir plus bas)

## Création du plugin vpype-rpgl

Les sources sont disponibles sur gitlab

D'après la documentation de pipx

1. Utilisation de cookiecutter pour faire un squelette de plugin vpype

```
# installer cookiecutter
pipx install cookiecutter
# initialiser un projet à partir du modèle de plugin
cookiecutter gh:abey79/cookiecutter-vpype-plugin #
```

Un formulaire permet de changer le noms du module (ici `vpype-rpgl`) et de la commande (ici `rpwrite`)

2. Conversion

Le processus a lieu dans le fichier `vpype_rpgl/rpwrite.py`

Inspiré du plugin vpype-gcode

```
from __future__ import annotations
import click
import copy

import vpype as vp
import vpype_cli


def cplx(p: complex, offset_x, offset_y) -> str:
    return f"{int(round(p.real)+offset_x)},{int(round(p.imag)+offset_y)}"

# @click.command()
@click.argument("output", type=vpype_cli.FileType("w"))
@vpype_cli.cli.command(group="Plugins")
@vpype_cli.global_processor
def rpwrite(document: vp.Document, output: typing.TextIO) -> vp.LineCollection:
    """
    Write rp-gl2 files for the vpype pipeline.
    """
    orig_document = document
    document = copy.deepcopy(document)  # do NOT affect the pipeline's document
    unit_scale = vp.convert_length('mm')
    # unit_scale = 100
    offset_x = -17750
    offset_y = -11180
    mult = 50 / 1.25
    document.scale(mult / unit_scale, mult / unit_scale)
    lc = vp.LineCollection()
    layers = document.layers
    output.write('IN;')
```

```
    output.write('PA;')
    output.write('SP1;PU;\n')
    output.write('OS;\n')
    for (layer_id, layer) in layers.items():
        for line in layer:
            output.write('PU' + cplx(line[0], offset_x, offset_y) + ';\n')
            for pt in line[1:]:
                output.write('PD' + cplx(pt, offset_x, offset_y) + ';\n')

    output.write('PU 0,0 ;\n')
    return orig_document
```

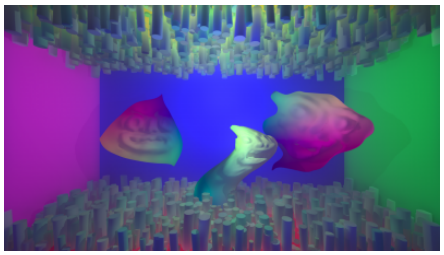
### TODO

- Corriger les trucs qui ne marchent pas cités plus haut
- Ajoutes des paramètres (vitesse et pression) voir des profils machines comme pour vpype-gcode

## IHNMAIMS

Deux triangles. Tests de plusieurs fonctionnalités 3D dans le moteur de jeu Godot :

- importation d'objets 3D depuis Blender au format gltf (recommandé dans la doc de Godot) → workflow pas super fluide quand on veut modifier les shaders appliqués aux matériaux des objets importés ;
- utilisation de multimeshes → plusieurs milliers d'instances d'un mesh avec des paramètres de shader différents ;
- utilisation de global illumination avec la technique de VoxelGI ;
- importation et utilisation de blend shapes ;



Article extrait de : http://www.lesporteslogiques.net/wiki/ - **WIKI Les Portes Logiques**
Adresse : **http://www.lesporteslogiques.net/wiki/recherche/residence_polygones/start?rev=1763479824**
Article mis à jour: **2025/11/18 16:30**