

# Polygones dégénérés

Au fablab @ Flux, 8-12 novembre 2025

Pad : [https://annuel.framapad.org/p/rda\\_polygone\\_2025](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 avec : <https://castle-engine.io/>

<https://www.qiew.org/>

<https://web.archive.org/web/20140412054654/http://cic.nist.gov/vrml/vbdetect.html>

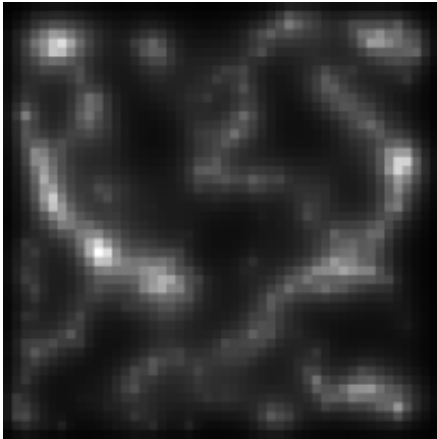
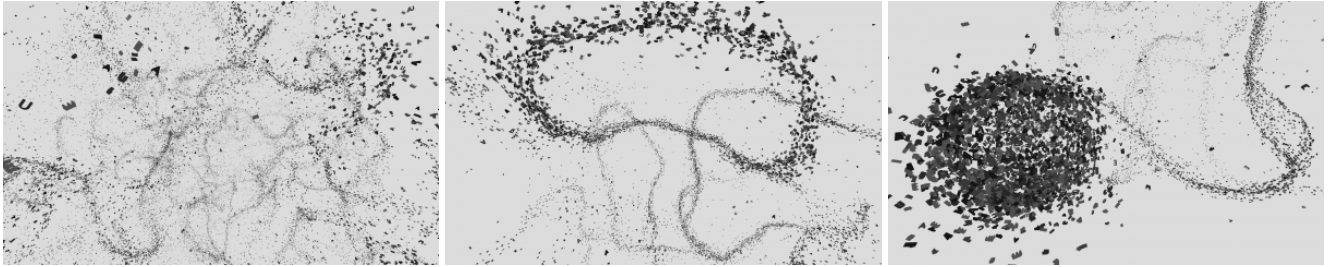
# Physarum Polycephalum

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 (saupoudré de polygones, histoire de...).

- Utilisation d'un compute shader pour accélérer le calcul de diffusion des traces chimiques.
- Grille 3D relativement petite (32x32x32 à 100x100x100) avec un nombre d'agent/particules allant de 5000 à 20.000

(code source à venir)



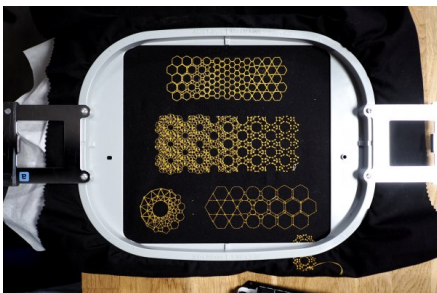
(vue de coupe au centre de la grille de voxels)

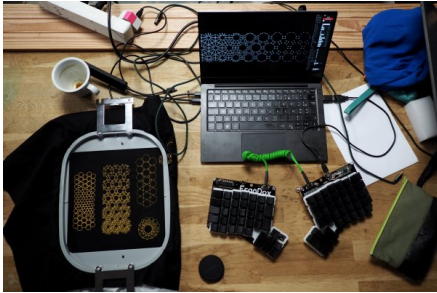
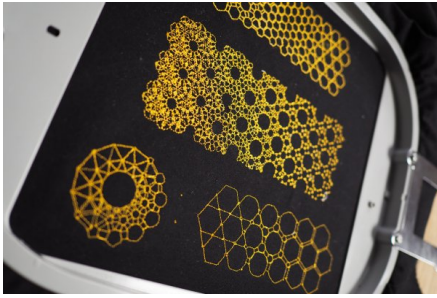
## Travaux similaires (et plus aboutis)

- [PolyPhy](#)
- [PhysOM](#)

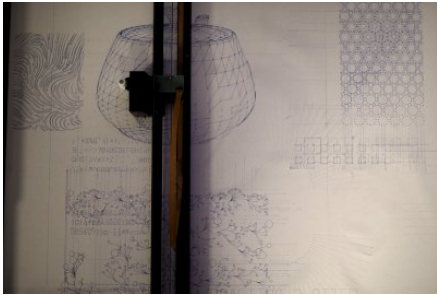
## Pavages de polygones brodés

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 vtype-rpgl

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

## Usage

```
vptype read input.svg rpwrite output.rppl
```

## Streamer le fichier avec hplot

### Installation [hplot](#)

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

## Usage

```
hplot -f query -B 32 /dev/ttyUSB0 output.rppl
```

## Les trucs qui ne marchent pas

- Liste à puce Le dessin s'arrête avant la fin, il manque les quelques derniers traits et souvent le stylo reste baissé.
- Liste à puce Le dessin est retourné en y (le haut est en bas) ⇒ à corriger dans vptype-rppl
- 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 vptype-rppl

Les sources sont disponibles sur [gitlab](#)

D'après [la documentation de pipx](#)

### 1. Utilisation de cookiecutter pour faire un squelette de plugin vptype

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

Un formulaire permet de changer le noms du module (ici vptype- rppl) et de la commande (ici rpwrite)

### 2. Conversion

Le processus a lieu dans le fichier vptype\_rppl/rpwrite.py

Inspiré du plugin [vptype-gcode](#)

```
from __future__ import annotations
import click
import copy

import vptype as vp
import vptype_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=vptype_cli.FileType("w"))
# @vptype_cli.cli.command(group="Plugins")
# @vptype_cli.global_processor
def rpwrite(document: vp.Document, output: typing.TextIO) -> vp.LineCollection:
    """
    Write rp-gl2 files for the vptype 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 vptype-gcode

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

Adresse : [http://www.lesporteslogiques.net/wiki/recherche/residence\\_polygones/start?rev=1762905114](http://www.lesporteslogiques.net/wiki/recherche/residence_polygones/start?rev=1762905114)

Article mis à jour: **2025/11/12 00:51**