mesh 2 svg 2 paper

Meshlab : https://www.meshlab.net/ Rien tiré de meshlab pour transformer un mesh (stl, obj) en svg

Premier essai concluant avec https://www.svgai.org/convert/stl-to-svg, le fichier s'ouvre bien avec inkscape, l'épaisseur des traits est bien trop élevée mais ça s'arrange facilement. Aucune face n'est cachée

Conseil de Laurent : utiliser «ln» de Michael Fogleman : https://github.com/fogleman/ln C'est programmé en Go, jamais utilisé

Conversion de formats 3D en ligne de commande

Avec OpenCTM (https://sourceforge.net/projects/openctm/)

```
sudo apt install openctm-tools
```

Ensuite on peut utiliser **ctmconv** qui permet de convertir les formats suivants :

- OpenCTM (.ctm),
- Stanford triangle format (.ply),
- Stereolitography (.stl),
- 3D Studio (.3ds),
- COLLADA 1.4/1.5 (.dae),
- Wavefront geometry file (.obj),
- LightWave object (.lwo),
- Geomview object file format (.off),
- VRML 2.0 - export only (.wrl).

Installation de Go

```
# ************************************     # installation du langage Go sur Debian 12  @ tenko
sudo apt update
sudo apt install golang
go version                                 # go version go1.19.8 linux/amd64
go env GOPATH                              # ok : /home/emoc/go
```

Helloworld en Go

Créer un fichier vide helloworld.go

```
nano helloworld.go
```

Le fichier helloworld.go contient

```
package main

import "fmt"

func main() {
    fmt.Println("HelloWorld, Golang!")
}
```

Puis

```
go run hello.go
```

Comment compiler ce programme pour qu'il puisse être utilisé comme une commande ?

Il faut le transformer en module

```
go mod init example/helloworld      # donner un nom et chemin au module
go mod tidy                         # récupérer les dépendances
```

```
go build -o helloworld              # créer le binaire «helloworld»
mv ./helloworld ../bin/helloworld
```

Maintenant on peut déclencher la commande avec

```
~/go/bin/helloworld
```

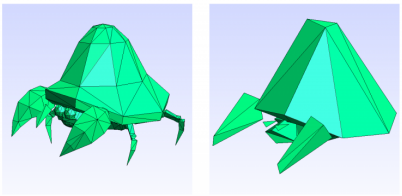
Utilisation de Simplify

Simplify est un logiciel en ligne de commande de Michael Fogleman qui permet de réduire le nombre de faces d'un objet 3D. Simplify est programmé en Go

https://github.com/fogleman/simplify

```
# installer Go (voir ci-dessus)
mkdir ~/go/bin
go install github.com/fogleman/simplify/cmd/simplify@latest
# réduction à 10% des faces de l'objet (652 faces -> 64 faces)
~/go/bin/simplify -f 0.1 parasect.stl parasect-0.1.stl
```

Comparaison (objet original : parasect)



Utilisation de ln

Pour transformer un objet 3D au format .OBJ en fichier .SVG

```
git clone https://github.com/fogleman/ln.git
cd ln
go mod init ln/ln
go mod tidy
```

placer le fichier teapot.obj dans le dossier et créer le fichier teapot.go :

```
package main

import "github.com/fogleman/ln/ln"

func main() {
    scene := ln.Scene{}
    mesh, err := ln.LoadOBJ("teapot.obj")
    if err != nil {
        panic(err)
    }
    mesh.UnitCube()
    scene.Add(ln.NewTransformedShape(mesh, ln.Rotate(ln.Vector{0, 1, 0}, 0.5)))
    // scene.Add(mesh)
    eye := ln.Vector{-0.5, 0.5, 2}
    center := ln.Vector{}
    up := ln.Vector{0, 1, 0}
    width := 1024.0
    height := 1024.0
    paths := scene.Render(eye, center, up, width, height, 35, 0.1, 100, 0.01)
    paths.WriteToPNG("teapot.png", width, height)
    paths.WriteToSVG("teapot.svg", width, height)
}
```

Puis

```
go run teapot.go
```

Ça marche! Le fichier svg est créé, en fonction du point de vue défini dans le script go, les faces qui doivent l'être sont cachées.

Transformer en exécutable. La commande est lancée depuis le répertoire courant dans lequel se trouve le fichier teapot.obj, les fichiers résultants (teapot.png et teapot.svg) sont créés dans le répertoire courant.

```
go build -o teapot           # construire le binaire
mv teapot ../bin/teapot      # déplacer dans le dossier ~/go/bin
~/go/bin/teapot              # lancer la commande depuis le répertoire courant
```