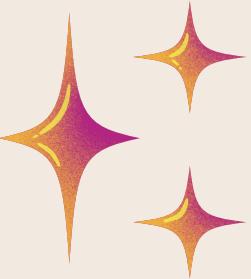


# PROYECTO FINAL

## IMPRESORA 3D

INTEGRANTES:

CORONADO MATEO  
QUIZHPE EDISON  
SANI NICOLE  
ZÚÑIGA LIAM



# INTRODUCCIÓN

- > La impresora 3D fue creada en 1981 por el japonés Hideo Kodama
- > Deposita materiales como plásticos, resinas, metales o termoplásticos en una serie de capas sucesivas con el cabezal que trabaja de abajo hacia arriba
- > Se programa el recorrido que este debe seguir gracias al conjunto de instrucciones que el usuario generaría de acuerdo con el diseño que se quiera implementar
- > Simular el comportamiento de esta impresora pero en un entorno 2D basándonos en un archivo proporcionado de SVG y parámetros específicos que se pueden configurar



# DESARROLLO MATEMÁTICO

> Creación de mallas

$$x_i = x_{min} + i * d_x$$

$$y_j = y_{min} + j * d_y$$

$x_{min}$  = Coordenada mínima en el eje X

$y_{min}$  = Coordenada mínima en el eje y

$d_x$  = son las distancias entre puntos en el eje x (resolución de impresión)

$d_y$  = son las distancias entre puntos en el eje y (resolución de impresión)

$i$  = Índice en la dirección X

$j$  = Índice en la dirección y

# DESARROLLO MATEMÁTICO

> Punto en un polígono

*Dado un polígono  $P$  definido por vértices  $(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)$*

*Un punto  $P$  se considera dentro si:*

$$C(x, y) = \sum_{i=1}^n I((x_{-i}, y_{-i}), (x_{i+1}, y_{i+1}), (x, y))$$

# LIBRERÍAS

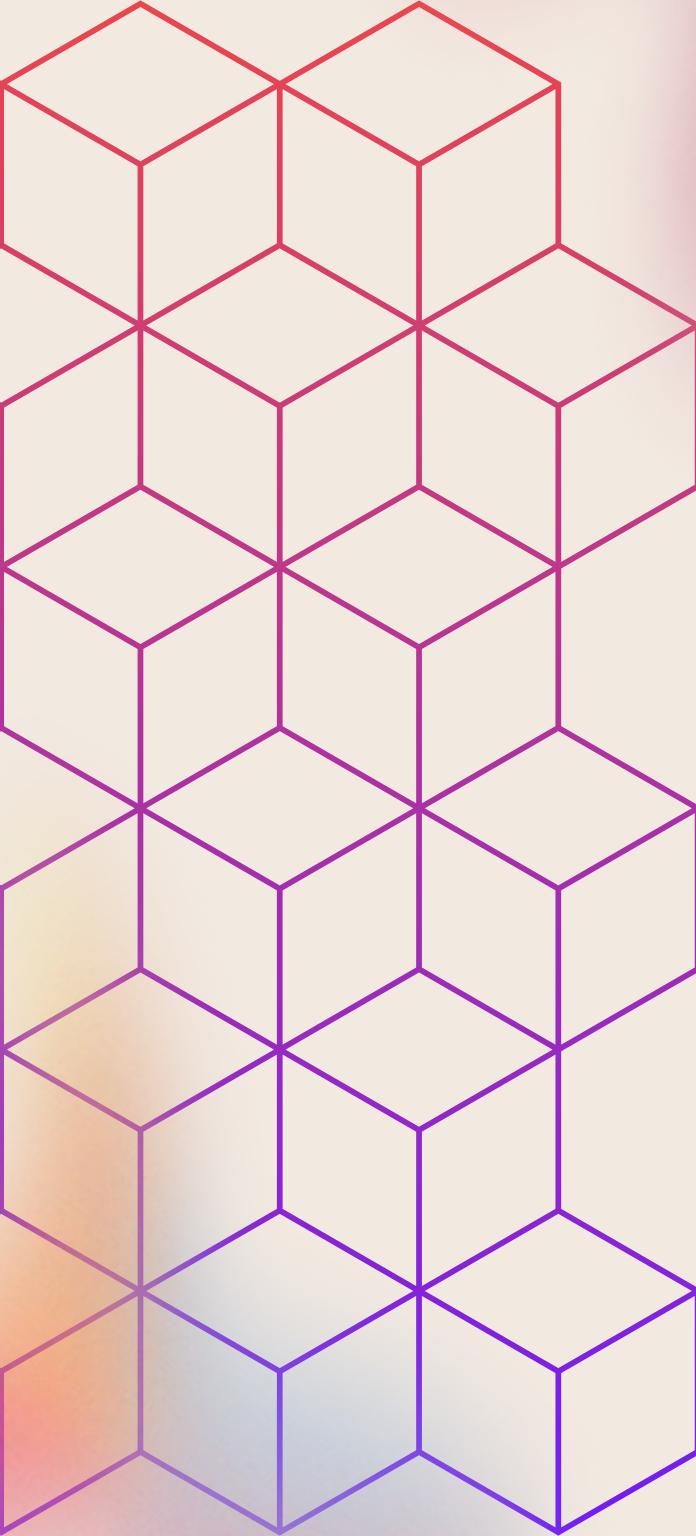
```
import matplotlib.patches as patches  
import tkinter as tk
```

```
from PIL import Image, ImageTk  
import matplotlib.pyplot as plt  
import matplotlib.animation as animation  
from tkinter import messagebox, colorchooser  
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg  
from matplotlib.path import Path
```

Este código implementa una aplicación gráfica en Python utilizando Tkinter y Matplotlib. Su objetivo es cargar archivos SVG, extraer sus contornos y simular el proceso de impresión 3D desde la parte superior, haciendo que se observe en dos dimensiones.

```
import numpy as np
import svgpathtools
import matplotlib.pyplot as plt
import matplotlib.animation as animation
from tkinter import messagebox, colorchooser
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
from matplotlib.path import Path
import time
```

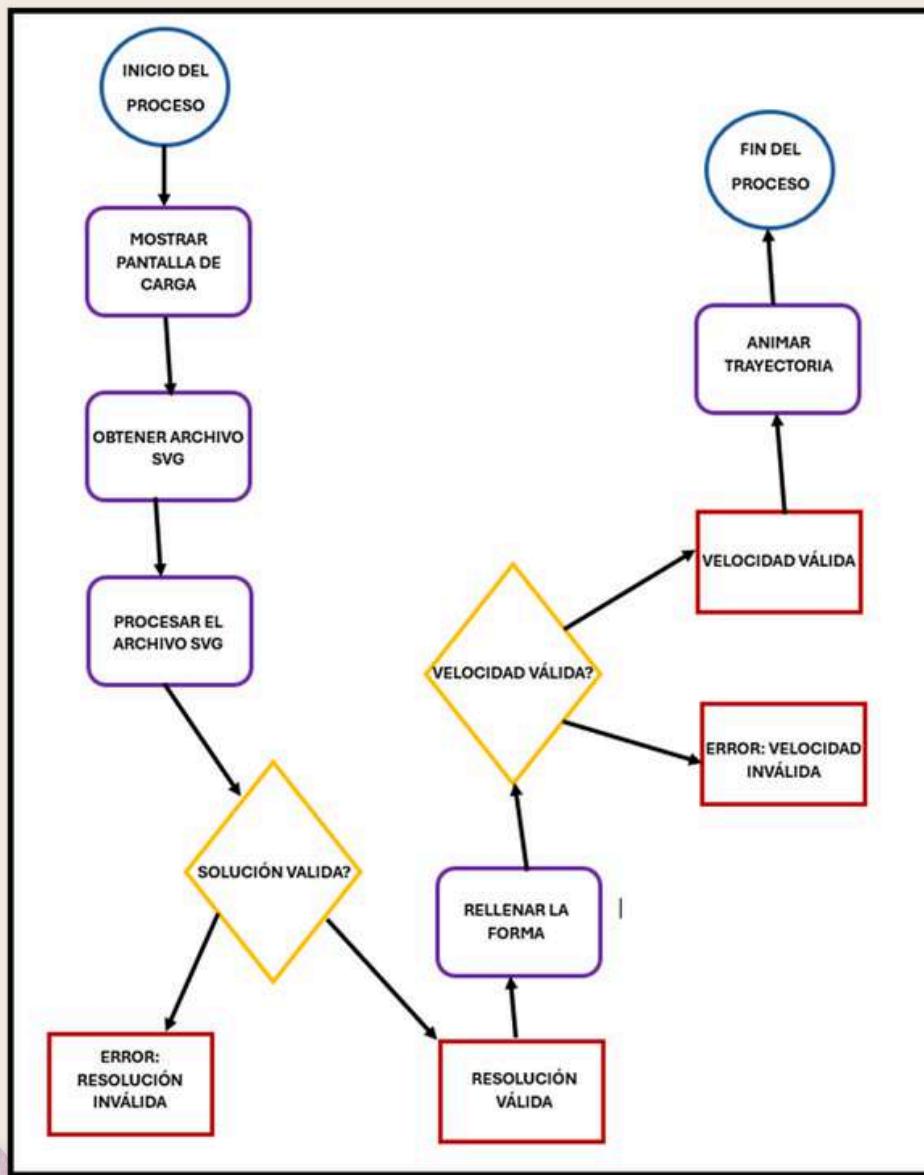
- tkinter
- numpy
- svgpathtools
- matplotlib.pyplot
- matplotlib.animation
- time
- tkinter.messagebox y tkinter.colorchooser
- matplotlib.backends.backend\_tkagg
- matplotlib.path.Path



# FUNCIONES IMPLEMENTADAS

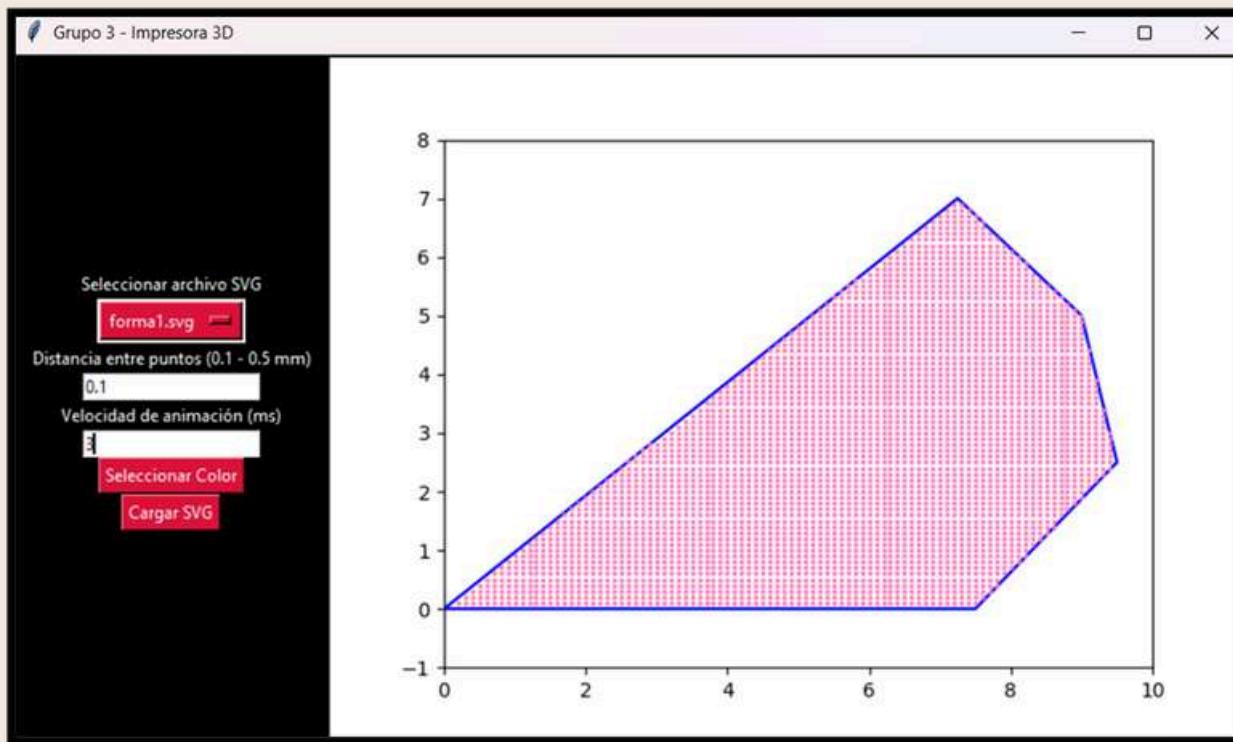
- Show\_loading\_screen()
- Load\_svg()
- Process\_svg(paths, file\_name)
- Validate\_resolution()
- Validate\_speed()
- Choose\_color()
- Fill\_shape(contour, file\_name)
- Animate\_trajectory(contour, filling\_points, x\_range, y\_range)
- Start\_ui()

# DIAGRAMA DE FLUJO



# CONCLUSIÓN

- El código está estructurado con funciones independientes para cada tarea específica, y a su vez están documentadas, lo que mejora su legibilidad, mantenimiento y escalabilidad. Esto permite futuras mejoras o modificaciones sin afectar la funcionalidad central.
- El software permite modificar parámetros clave, como la resolución de impresión y la velocidad de animación, lo que ayuda a evaluar diferentes configuraciones antes de una impresión real.





# GRACIAS

