



TRABAJO FIN DE GRADO  
INGENIERÍA INFORMÁTICA

# Desarrollo de herramientas para facilitar el diseño de piezas en máquinas de corte láser

**Autor**

Pedro Luis Hurtado González

**Tutor**

Sergio Alonso Burgos



Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación  
—  
Granada, septiembre de 2017









# **Desarrollo de herramientas para facilitar el diseño de piezas en máquinas de corte láser**

## **Autor**

Pedro Luis Hurtado González

## **Tutor**

Sergio Alonso Burgos



# **Desarrollo de herramientas para facilitar el diseño de piezas en máquinas de corte láser**

Pedro Luis Hurtado González

**Palabras clave:** corte láser, Inkscape, SVG, lxml, python, extensión, sólidos platónicos, UGR.

## **Resumen**

El objetivo de este proyecto se centra en generar una documentación bien estructurada de como crear extensiones para Inkscape y también diseñar e implementar ciertas extensiones o funciones para facilitar el trabajo a gente que utilice dicho programa para el diseño de piezas que serán cortadas por una cortadora láser.

Todas las extensiones que se creen se pondrán al alcance de quien las solicite, en la plataforma GitHub, y serán de código abierto, por tanto, todo aquel que quiera modificar el código para personalizarlo o acercarlo más a sus necesidades podrá hacerlo.



# **Free software development to facilitate design of pieces in laser cut machines**

Pedro Luis Hurtado González

**Key words:** laser cut, Inkscape, SVG, lxml, python, extension, platonic solids, UGR.

## **Summary**

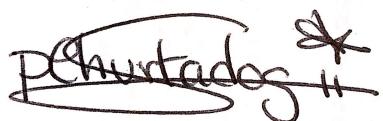
The main purpose of this project is to create a well structured documentation about how to create new Inkscape extensions. Additionally, some extensions or functions will be designed and implemented in order to facilitate the work of people that use this program to design pieces to be cut by a laser cut machine.

Every created extension will be made freely available to whoever requests it and the extensions will be offered as free software, thus anyone who wishes to modify the code in order to personalize it or tailor it to their specific needs will be able to do it.



---

Yo, **Pedro Luis Hurtado González**, alumno de la titulación de grado en ingeniería informática de la **Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación de la Universidad de Granada**, con DNI 50616802N, autorizo la ubicación de la siguiente copia de mi Trabajo Fin de Grado en la biblioteca del centro para que pueda ser consultada por las personas que lo deseen.

A handwritten signature in black ink. It starts with a large 'P' inside an oval, followed by 'ehurtado' and a small star-like symbol at the end.

Fdo: Pedro Luis Hurtado González

Granada a 12 de septiembre de 2017



---

D. **Sergio Alonso Burgos**, Profesor del Departamento de Lenguajes y Sistemas informáticos de la Universidad de Granada.

**Informa:**

Que el presente trabajo, titulado *Desarrollo de herramientas para facilitar el diseño de piezas en máquinas de corte láser*, ha sido realizado bajo su supervisión por **Pedro Luis Hurtado González**, y autorizo la defensa de dicho trabajo ante el tribunal que corresponda.

Y para que conste, expido y firmo el presente informe en Granada a 12 de septiembre de 2017.

**El tutor:**

A handwritten signature in blue ink that reads "Sergio A." The "S" and "A" are capitalized and slanted, with a horizontal line underneath the "S" and "A".

Fdo: **Sergio Alonso Burgos**





# **Agradecimientos**

Dedico este Trabajo Fin de Grado a todos los docentes capaces de transmitir entusiasmo por el aprendizaje y, en particular, a mi tutor Sergio Alonso porque su buen criterio y continua disponibilidad han sido indispensables para que este proyecto llegue a buen puerto.

También quiero dedicarlo a mi familia, por sus muchos años de entrega y los consejos sobre la importancia de la Educación en el desarrollo de las personas.

Por último, pero no menos importante, me gustaría tener presente a los compañeros y compañeras de estudio - he aprendido tanto con ellos y de ellos -, que siempre formarán parte de lo que soy.





# ÍNDICE

<b>1. INTRODUCCIÓN Y MOTIVACIÓN</b>	<b>22</b>
1.1. ¿Qué es el corte láser?	22
1.2. CNC Y G-CODE	22
1.3. Gráficos vectoriales y rasterizados	23
1.4. SVG	23
1.5. Inkscape	24
1.6. Licencia GPL/GNU	24
1.7. Asignaturas aplicadas en el proyecto	25
1.8. Motivación	25
<b>2. OBJETIVOS</b>	<b>27</b>
2.1. Obligatorios	27
2.2. Opcionales	30
<b>3. METODOLOGÍA DE DESARROLLO Y ESTIMACIONES</b>	<b>32</b>
3.1. Estimación temporal	32
3.2. Estimación económica	36
<b>4. DOCUMENTACIÓN PARA CREAR NUEVAS EXTENSIONES</b>	<b>38</b>
4.1. Comunicación entre Inkscape y extensiones	38
4.2. Estructura interna de ficheros SVG	39
4.3. Archivos xml .inx	41
4.4. Archivos python .py	43
4.5. LXML (Element)	44
4.6. Inkex.py	45
4.7. Simplestyle.py	46
4.8. Línea de comandos	46
4.9. Instalación de extensiones	49
4.10. Ejemplo	49
<b>5. EXTENSIONES</b>	<b>54</b>
5.1. Capa por color	54
5.2. Color por capa	59
5.3. TabbedBoxMaker	65
5.4. Sólidos platónicos	69
5.5. Ajuste sobre letras de un solo trazo	79
5.6. Patrón para doblar madera	81

5.7. Exportar capas a EPS	85
<b>6. CONCLUSIÓN Y TRABAJOS FUTUROS</b>	<b>91</b>
<b>7. ANEXO: PARTES DE LAS REUNIONES CON EL CLIENTE</b>	<b>94</b>
7.1. Parte 1	94
7.2. Parte 2	95
7.3. Parte 3	95
7.4. Parte 4	96
7.5. Parte 5	96
7.6. Parte 6	97
7.7. Parte 7	97
<b>8. REFERENCIAS</b>	<b>99</b>



# 1. INTRODUCCIÓN Y MOTIVACIÓN

Como su propio nombre indica, este proyecto está orientado a la creación de ciertas herramientas de manera que se les facilite el trabajo a personas que utilizan Inkscape para diseñar los objetos que serán cortados por una cortadora láser.

Todos los archivos de código generados están alojados en la plataforma GitHub para que cualquier usuario pueda utilizarlos. <https://github.com/plhurtado/CortadoraLaser>

Hay varios conceptos importantes a la hora de documentar y entender este proyecto por lo que, en lo que sigue, se explicarán de manera más cercana.

---

## 1.1. ¿Qué es el corte láser?

El corte láser<sup>[1]</sup> es una tecnología que utiliza un láser para cortar materiales, se utiliza normalmente en empresas industriales y más recientemente se ha comenzado a utilizar en escuelas, pequeñas empresas y como hobby.

El funcionamiento general de este sistema consiste en dirigir un haz láser de alta potencia a través de unas lentes hacia el material a cortar. A partir de aquí también tenemos que hablar de CNC o control numérico por computadora que es el responsable de dirigir el láser por la superficie del material o de mover el material bajo el haz láser.

Una vez que el láser entra en contacto con el material a cortar dicho material se funde, quema o vaporiza por donde ha ido pasando el láser dejando un corte con un acabado de alta calidad en la mayoría de casos. Normalmente este sistema se utiliza para cortar láminas de chapa o madera aunque cada vez se usa para cortar nuevos materiales como cuero, tela o metal.

---

## 1.2. CNC Y G-CODE

Como se ha comentado en la introducción del corte láser CNC<sup>[2]</sup> significa control numérico por computadora y es un sistema que permite controlar en todo momento la posición de un elemento físico, normalmente una herramienta que está montada en una máquina, que en nuestro caso sería el haz láser. Se podría decir que mediante un software y un conjunto de órdenes, controlaremos las coordenadas de posición de un punto (el foco láser) respecto a un origen (0,0,0) de máquina.

Las coordenadas del control numérico por computadora pueden estar formadas por dos valores, en caso de que sea movimiento en solo dos dimensiones, o por tres, en caso de que se pudiera controlar también la distancia a la que el rayo impacta sobre el material (esto se puede buscar en caso de que no se quiera tan solo cortar, si no que también se quieran hacer grabados o similares sobre el material).

CNC no controla tan solo los puntos por los que se desplazará el objeto en cuestión si no que también tiene variables para poder controlar la manera de desplazarse entre ellos, su velocidad y algunos parámetros más.

G-CODE<sup>[3]</sup> (conocido también como lenguaje de programación G) es un lenguaje de programación principalmente utilizado en la automatización industrial. Además forma parte de la

ingeniería asistida por computadora y es el lenguaje más usado en control numérico por computadora. Sus principales usos son en cortadoras láser, impresoras 3D e instrumentos de medición.

---

### 1.3. Gráficos vectoriales y rasterizados

Las imágenes rasterizadas<sup>[4]</sup> (o de mapa de bits) se describen mediante un conjunto de bits dentro de una cuadrícula rectangular de píxeles o puntos. Por otro lado, las imágenes vectoriales se describen mediante líneas, formas y otros componentes gráficos de imagen almacenados en un formato que incorpora fórmulas geométricas para interpretar los elementos de la imagen.

Imagen rasterizada (mapa de bits): Al 100%, la versión de mapa de bits o rasterizada de la imagen es prácticamente igual que la versión vectorial. Observe que en cuanto se redimensiona la versión rasterizada, los píxeles del borde comienzan a mostrarse y ya no se ven difuminados.

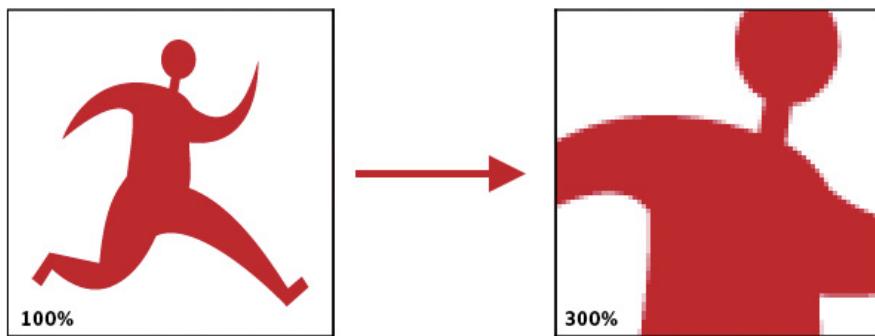
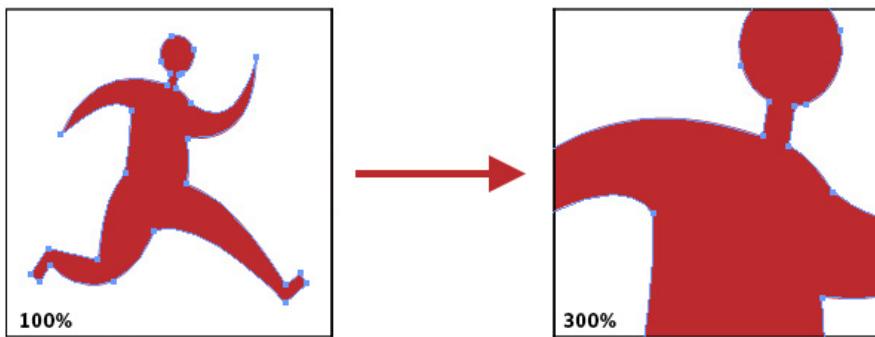


Imagen vectorial: La imágenes vectoriales se crean mediante la definición de puntos y curvas por lo que, cuando se redimensionan, los bordes se mantienen enfocados y nítidos, independientemente del tamaño.



---

### 1.4. SVG

Su traducción literal sería “Gráficos Vectoriales Escalares”<sup>[5]</sup>. Es un formato libre de documento donde se especifica un gráfico vectorial con gran facilidad para escalar, o lo que es lo mismo, para cambiar el tamaño de la imagen. Estos documentos están definidos en XML y permiten usar formas gráficas, mapas de bits o texto y pueden ser estáticos o dinámicos.

Aunque las primeras versiones no se podían ver en los diferentes navegadores, hoy ya es un estándar que funciona sin problemas en todos los navegadores. SVG se convirtió en una recomendación del [W3C](#) en septiembre de 2001 con lo que en estos momentos ya es admitido por todos.

---

## 1.5. Inkscape

Inkscape<sup>[6]</sup> es un editor de gráficos vectoriales de código abierto, similar a programas como *Adobe Illustrator*, *Corel Draw*, *Freehand* o *Xara X*. Lo que lo hace único es que usa como formato nativo el *Scalable Vector Graphics (SVG)*.

El nombre está compuesto por las dos palabras inglesas *ink* (tinta) y *scape* (paisaje o vista). La tinta es una sustancia común para dibujos, y se utiliza cuando el trabajo bocetado está listo para ser trasladado en forma permanente al papel; por lo tanto evoca la idea de que Inkscape está listo para trabajo de producción. Y *scape* es una vista de una gran cantidad de objetos, como un paisaje o vista al mar, y por lo tanto alude a la naturaleza orientada a objetos propia de las imágenes vectoriales.

Inkscape es una aplicación libre y se entiende con ello que es libre de coste, uso y distribución. Al ser de código abierto, nos permite crear diferentes extensiones o funciones de manera que se personalice lo máximo posible a nuestros gustos y necesidades.

El desarrollo de Inkscape se adhiere al estándar de código fuente abierto (*open source*) con la intención de proveer a la comunidad de usuarios de un producto sólido y útil. Dicho proceso de desarrollo es abierto y está basado en la comunidad, para crear un núcleo sólido y ampliable que parte del código de *Sodipodi Hydra*.

Inkscape provee paquetes binarios para Linux, Windows 2000/2003/XP/Vista/7/8/10 (autoinstalable completo), y Mac OS X (paquete dmg).

---

## 1.6. Licencia GPL/GNU

La licencia que he escogido para la implementación de este proyecto es GPL versión 3.0<sup>[7]</sup>.

La Licencia Pública General de GNU o GPL es la licencia de derecho de autor más ampliamente usada en el mundo del software libre y código abierto, y garantiza a los usuarios finales (personas, organizaciones, compañías) la libertad de usar, estudiar, compartir (copiar) y modificar el software.

Su propósito es doble: declarar que el software cubierto por esta licencia es libre, y protegerlo (mediante una práctica conocida como *copyleft*) de intentos de apropiación que restrinjan esas libertades a nuevos usuarios cada vez que el código o proyecto es distribuido, modificado o ampliado.

---

## 1.7. Asignaturas aplicadas en el proyecto

Durante el desarrollo de este proyecto me han sido especialmente útiles algunos conceptos o metodologías aprendidos en diferentes asignaturas del grado como:

- Metodologías para el desarrollo del software (Metodologías de desarrollo ágiles o Dirección y gestión de proyectos).
- Habilidades para aprender nuevos lenguajes de programación (Fundamentos de programación, metodología de programación, programación y diseño orientado a objetos o desarrollo de software)
- Planificación de proyectos (Dirección y gestión de proyectos)
- Ingeniería de requisitos (Fundamentos de ingeniería del software, Diseño y desarrollo de sistemas de información)

---

## 1.8. Motivación

A la hora de escoger plataforma, sobre la que desarrollar las extensiones, opté por Inkscape porque es una de las herramientas de gestión de gráficos vectoriales más importantes y utilizada del mercado en la actualidad y no contaba con una librería dedicada a las personas que utilizan máquinas de corte láser. También tuve en cuenta que este programa es muy completo y cuenta con muchos repositorios que me podrían facilitar el trabajo de desarrollo.

Además Inkscape trabaja con ficheros SVG que es el formato escogido por la mayoría de personas que trabajan con diseños 3D y corte láser.

Por supuesto, otro punto que me ayudó a decantarme por este programa fue que es de código abierto y que cualquier desarrollador puede acceder a los ficheros de código para usarlos como base o personalizarlo y adaptarlo a sus necesidades.



## 2. OBJETIVOS

El proyecto consiste en la creación de una librería de extensiones o funciones libres, orientadas a facilitar el diseño y modelado a la gente que utilice Inkscape para máquinas de corte láser.

Se podría decir que este proyecto está formado por varios subproyectos o tareas que serán cada una de las extensiones que se realicen y la documentación.

Las diferentes extensiones podrán ser de carácter obligatorio u opcional (indicado por el tutor) dependiendo de su dificultad y del tiempo requerido para su desarrollo.

Para cada una de las extensiones se tendrán que crear varios apartados de desarrollo del trabajo de manera obligatoria, estos son: Análisis, Diseño, Implementación y Pruebas.

No obstante algunas extensiones pueden contar con un apartado denominado Observaciones para anotaciones a tener en cuenta sobre la extensión en sí.

Cada una de las extensiones, ya sea de carácter obligatoria u opcional, se subdividirá en nuevos objetivos que pueden ser obligatorios u opcionales.

---

### 2.1. Obligatorios

#### 1. Documentación para crear nuevas extensiones:

Obligatorio:

- 1.1. Comunicación entre Inkscape y extensiones.
- 1.2. Estructura interna de ficheros SVG
- 1.3. Estructura de los ficheros xml (*.ink*).
- 1.4. Estructura de los ficheros python (*.py*) .

Opcional:

- 1.5. Descripción de LXML.
- 1.6. Descripción de la librería *inkex.py*.
- 1.7. Descripción de la librería *simplestyle.py*.
- 1.8. Guía sobre la línea de comandos.
- 1.9. Instalación de extensiones.

#### 2. Capas por color:

Obligatorio:

- 2.1. Diseñar e implementar una extensión que cree una capa nueva por cada uno de los colores diferentes que nos encontramos en el proyecto y cuelgue de estas capas los elementos en cuestión.
- 2.2. Si hay varios objetos del mismo color no se crearán varias capas, se meterán los dos en una única capa.

Opcional:

- 2.3. Poner como nombre de la capa el nombre del color que tienen los objetos en el borde (se buscará el nombre más adecuado en un diccionario estándar de colores).
- 2.4. Eliminar las capas originales que no se van a utilizar.
- 2.5. Se tendrán en cuenta las cajas3D. Se creará una capa para estos objetos de manera que se agrupen todas las partes de estas cajas en una única capa (también se agruparán).
- 2.6. Añadir una pestaña de ayuda a la extensión donde se explique como funciona.

3. Color por capa:

Obligatorio:

- 3.1. Diseñar e implementar una extensión que aplique el mismo color de borde a todos las figuras de una misma capa y este color sea diferente entre las distintas capas.
- 3.2. Dar la opción de eliminar todas las capas tras aplicarle el color a los bordes.

Opcional:

- 3.3. Escoger los colores de manera aleatoria entre el diccionario de colores que aparece en *simplestyle.py*.
- 3.4. Añadir el nombre del color que se ha puesto a los bordes al nombre de la capa, de manera que si la capa se llama “LAYER1” pase a ser “LAYER1 + red”, suponiendo que se haya aplicado el borde de color “red”.
- 3.5. Añadir una pestaña de ayuda a las extensión donde se explique como funciona y que configuración tiene el parámetro.

4. TabbedBoxMaker:

Obligatorio:

- 4.1. Modificar la extensión ya creada<sup>[8]</sup> para generar todos los trazos de una pieza en un único *path*.
- 4.2. Hacerlo de manera que sea configurable, es decir, que se pueda elegir desde la interfaz si se quiere todo en *paths* diferentes o un *path* por pieza.

5. Sólidos platónicos:

Obligatorio:

- 5.1. Diseñar e implementar una extensión generadora de sólidos platónicos.
- 5.2. Se tendrá que poder configurar el sólido platónico que se va a dibujar, la longitud de sus lados, el número de pestañas por lado, la anchura de las pestañas y el grosor del material.

Opcional:

- 5.3. Hacer que se puedan dar las medidas de los sólidos en diferentes unidades de medida.
- 5.4. Dar la opción de que se dibuje una única pieza.
- 5.5. Ofrecer dos opciones diferentes para la colocación de las piezas sobre el tablero de dibujo. Todos los objetos en línea o con formato rectangular.

- 5.6. Implementar una opción que nos permita generar las caras de los objetos de forma calada y que sea configurable la distancia entre la pared interna y externa.
- 5.7. Permitir que desde la interfaz gráfica se pueda seleccionar agrupar todos los objetos.
- 5.8. Se podrá configurar el color de trazo de las piezas y el grosor de éste.
- 5.9. La selección de color debería ser de una manera cómoda y no dando tres valores para rojo, verde y azul.
- 5.10. Añadir una pestaña de ayuda a la extensión donde se explique como funciona y que son cada uno de los parámetros.

6. Ajuste sobre letras de un solo trazo:

Obligatorio:

- 6.1. Diseñar e implementar una extensión que elimine el relleno de letras con fuentes orientadas a CNC y elimine los trazos que no son necesarios para hacerlas legibles.

Opcional:

- 6.2. Añadir una pestaña de ayuda a la extensión donde se explique como funciona.

7. Patrón para doblar madera:

Obligatorio:

- 7.1. Diseñar e implementar una extensión que dibuje un patrón de líneas con el que conseguimos poder doblar planchas de madera.
- 7.2. Para la correcta creación se darán una serie de atributos como la anchura del patrón, longitud, separación horizontal y vertical de las líneas, longitud máxima de las líneas y marcas iniciales.

Opcional:

- 7.3. Configurar la extensión para que las medidas se puedan dar en diferentes unidades de medida.
- 7.4. Dar la opción de agrupar las líneas.
- 7.5. Permitir escoger entre dibujar todas las líneas en un único *path* o un *path* por cada línea.
- 7.6. Permitir dibujar una caja contenedora con las medidas de anchura y altura que el usuario pasó como parámetro.
- 7.7. Se podrá configurar la distancia que se quiere que sobrepasen las líneas por arriba y por debajo de la altura máxima.
- 7.8. Se podrá escoger el orden de dibujado de las líneas. Se podrá seleccionar que se pinte siempre de arriba a abajo o que se vaya haciendo en zigzag.
- 7.9. Añadir una pestaña de ayuda a la extensión donde se explique como funciona y que son cada uno de los parámetros.

8. Exportar capas a EPS:

Obligatorio:

- 8.1. Diseñar e implementar una extensión capaz de exportar cada una de las capas a un archivo EPS diferente.
- 8.2. Se dará la opción de configurar la opacidad de los bordes de los objetos al 100%, de eliminar el relleno de las figuras y de pasar los textos a *paths* para que se configure antes de exportarlo.

Opcional:

- 8.3. Se podrá configurar el directorio donde se desean crear los archivos EPS.
- 8.4. Los archivos exportados tendrán el mismo nombre que las capas de proyecto.
- 8.5. Añadir una pestaña de ayuda a la extensión donde se explique como funciona y que son cada uno de los parámetros.

---

## 2.2. Opcionales

### 9. Colocación de los objetos del proyecto para aprovechar mejor el material sobre el que vamos a cortar.

Diseñar e implementar una extensión que estudie las formas geométricas de las figuras del proyecto para poder colocarlas de forma que se desperdicie la menor cantidad de material posible.

### 10. Colocación de los objetos del proyecto para optimizar el tiempo de corte.

Diseñar e implementar una extensión que coloque los objetos en el proyecto de manera que la cortadora láser recorra el menos espacio posible sin cortar para optimizar el tiempo de corte general.

### 11. Internacionalización del proyecto

Para conseguir que el proyecto sea internacional y se pueda utilizar en cualquier parte del mundo se pondrán todos los comentarios, documentación, nombre de variables, etc en inglés.



### 3. METODOLOGÍA DE DESARROLLO Y ESTIMACIONES

Como metodología de trabajo no se puede decir que haya seleccionado ninguna de las habituales, ya que al ser una única persona en vez de un equipo y trabajar con el tutor cumpliendo con la función de varios roles, no he podido ajustarme a todas las “reglas básicas” o patrones de ninguna de las metodologías existentes.

Sin embargo, el desarrollo de este proyecto lo he basado en metodologías ágiles en las cuales el equipo de trabajo mantiene de manera constante la comunicación con el cliente (en este caso el tutor) y se citan reuniones periódicamente cada poco tiempo.

Con la metodología ágil que más puntos comparto es la denominada SCRUM<sup>[9]</sup>, ya que cumple los siguientes requisitos:

- *Product Backlog*: Que es el conjunto de requisitos denominados historias descritos en un lenguaje no técnico y priorizados por valor de negocio o beneficio y coste aportados, en este caso, por el tutor.
- *Sprint planning*: De manera personal decido cual va a ser el conjunto de historias que se va a poder realizar en el Sprint actual y decido y organizo como voy a conseguirlo.
- *Sprint*: Tiempo de desarrollo e implementación de las historias.
- *Demo y retrospectiva*: Reunión que se realiza al final de cada Sprint con el tutor donde se muestra el trabajo finalizado asignado a dicho intervalo de tiempo y se hace una retrospectiva sobre los puntos que se deben fortalecer y cuales han de ser modificados para mejorar.

\*Esta metodología de trabajo cuenta con varios roles diferentes pero está orientada a grupos de trabajo por lo que en este caso no se ha podido definir un buen reparto de roles.

Aunque el desarrollo del proyecto se haga siguiendo una metodología ágil, se hará una planificación inicial a la cual me deberé adaptar lo máximo posible intentando simular que fuese un encargo de un proyecto real por parte de alguna empresa o particular.

#### 3.1. Estimación temporal

El proyecto lo he dividido en 9 grandes bloques, cada uno con ciertos subapartados para los cuales he estimado un tiempo concreto:

- Planificación, investigación y aprendizaje inicial:

Tareas	Estimación temporal
Tutorías iniciales	3 horas
Investigación sobre metodología	2 horas
Documentación tutorías	1 hora
Aprendizaje Python	6 horas
Documentación sobre LXML, inkex, simplestyle y command line	1 hora
Primeras pruebas (investigación)	2 horas
<b>TOTAL</b>	<b>15 horas</b>

- Capa por color:

Tareas	Estimación temporal
Ingeniería de requisitos	2 horas
Diseño	1 hora
Implementación	3 horas
Pruebas y correcciones	3 horas
Documentación	1 hora
<b>TOTAL</b>	<b>10 horas</b>

- Color por capa:

Tareas	Estimación temporal
Ingeniería de requisitos	2 horas
Diseño	1 hora
Implementación	3 horas
Pruebas y correcciones	3 horas
Documentación	1 hora
<b>TOTAL</b>	<b>10 horas</b>

- TabbedBoxMaker:

Tareas	Estimación temporal
Ingeniería de requisitos	1/4 hora
Diseño	1/2 hora
Implementación	1/2 hora
Pruebas y correcciones	1/2 hora
Documentación	1/4 hora
<b>TOTAL</b>	<b>2 horas</b>

- Sólidos platónicos:

Tareas	Estimación temporal
Ingeniería de requisitos	2 horas
Diseño	4 horas
Implementación	7 horas
Pruebas y correcciones	5 horas
Documentación	2 horas
<b>TOTAL</b>	<b>20 horas</b>

- Ajuste sobre letras de un solo trazo:

Tareas	Estimación temporal
Ingeniería de requisitos	1/2 hora
Diseño	1 hora
Implementación	2 horas
Pruebas y correcciones	1 hora
Documentación	1/2 hora
<b>TOTAL</b>	<b>5 horas</b>

- Patron para doblar madera:

Tareas	Estimación temporal
Ingeniería de requisitos	1/2 hora
Diseño	1 hora
Implementación	2 horas
Pruebas y correcciones	1 hora
Documentación	1/2 hora
<b>HORAS</b>	<b>5 horas</b>

- Exportar EPS:

Tareas	Estimación temporal
Ingeniería de requisitos	1 hora
Diseño	2 horas
Implementación	3 horas
Pruebas y correcciones	3 horas
Documentación	1 hora
<b>TOTAL</b>	<b>10 horas</b>

- Documentación final:

Tareas	Estimación temporal
Documentación final	5 horas

En total se estimó que para la realización del proyecto se invertirían alrededor de 82 horas.

Estas tareas se realizarían entre febrero y julio y se planificaron con la siguiente organización semanal:



### 3.2. Estimación económica

Para la estimación de presupuesto se han tenido en cuenta varios puntos valorándolos y calculando un total.

	<b>Unidades</b>	<b>Precio por unidad</b>	<b>Subtotal</b>
<b>Horas de trabajo</b>	82 horas	15 €/hora	1230 €
<b>Licencia GPL</b>	1	0	0 €
<b>Equipo informático (1)</b>	1 ordenador	1,85 €/hora	151,7 €
<b>Materiales para pruebas (2)</b>	5 planchas	15 €	7,5 €
		<b>TOTAL:</b>	<b>1389,2 €</b>

- (1) Se ha estimado que el precio del ordenador ronda los 2000€ y se cree que tendrá una vida útil de unos 3 años por lo que una hora de su uso supone 1,85 euros, que multiplicado por las 82 horas que ha sido utilizado da un total de 151,7€.
- (2) Materiales para pruebas incluye las planchas de madera (DM[10]) necesitadas para probar las extensiones.



## 4. DOCUMENTACIÓN PARA CREAR NUEVAS EXTENSIONES

En este apartado me centraré en el primer objetivo de este proyecto que consiste en generar una documentación estructurada y completa de forma que los programadores noveles en este campo puedan utilizarlo de base.

### 4.1. Comunicación entre Inkscape y extensiones

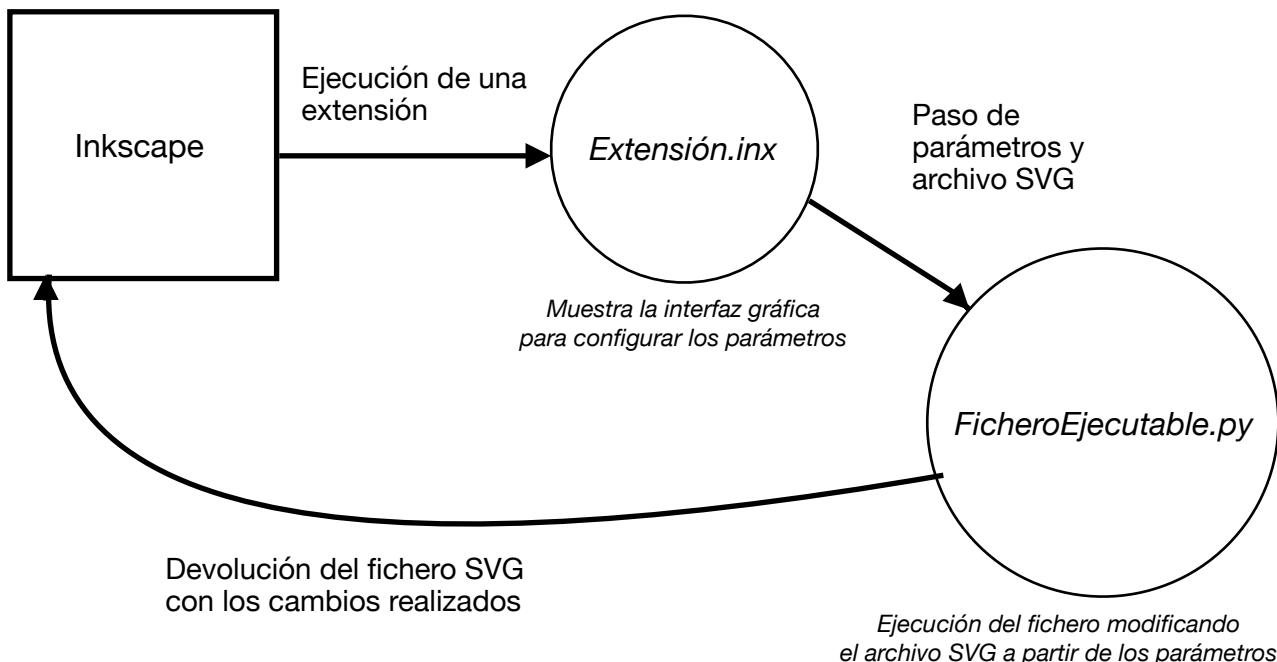
Al ejecutar una extensión desde Inkscape lo primero que aparece en la pantalla, en la mayoría de los casos, es una interfaz gráfica desde donde podremos configurar los diferentes parámetros o incluso leer en qué consiste la extensión en caso de que cuente con una pestaña de “ayuda”.

Se ha querido señalar que no en todas las extensiones vemos una interfaz gráfica porque hay casos en los que el programa ejecutable no necesita parámetros, tan solo necesita el proyecto parseado para modificarlo.

Estas interfaces están implementadas en XML.

Una vez que se han configurado los parámetros, se empaquetan y se envían, junto al documento SVG del proyecto parseado, al archivo ejecutable que será el encargado de generar los cambios sobre el proyecto para después volverlo a empaquetar y devolverlo a Inkscape. Cuando Inkscape recupera el archivo parseado del proyecto con los cambios, aplica las modificaciones sobre la interfaz gráfica y así el usuario puede ver las modificaciones aplicadas.

Los archivos ejecutables pueden estar implementados en cualquier lenguaje de programación, en mi caso siempre he utilizado python, pero permite ejecutar ficheros en c, c++, java, javaScript, etc.



## 4.2. Estructura interna de ficheros SVG

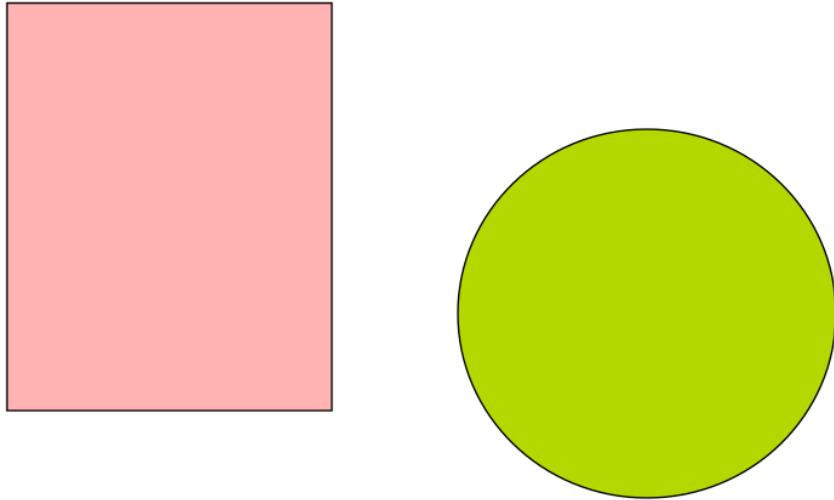
Toda la información referente a cualquier proyecto de Inkscape se almacenará en formato SVG basado en XML, como ya se dijo anteriormente. Para poder ver la estructura que siguen los proyectos tan solo tenemos que abrir el fichero con algún editor de texto plano y nos mostrará la estructura.

Un ejemplo de documento SVG muy sencillo sería el siguiente:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<! -- Created with Inkscape (http://www.inkscape.org/) -->

<svg
  xmlns:svg="http://www.w3.org/2000/svg"
  xmlns="http://www.w3.org/2000/svg"
  xmlns:sodipodi="http://sodipodi.sourceforge.net/DTD/sodipodi-0.dtd"
  xmlns:inkscape="http://www.inkscape.org namespaces/inkscape"
  width="210mm"
  height="297mm"
  id="svg2"
  >
<defs
  id="defs4" />
<sodipodi:namedview
  id="base"
  pagecolor="#ffffff"
  bordercolor="#666666"
  borderopacity="1.0"
  inkscape:document-units="px"
  inkscape:current-layer="layer1"
  showgrid="false"
  inkscape:window-width="1920"
  inkscape:window-height="979" />
<metadata
  id="metadata7">
  <rdf:RDF>
    <cc:Work
      rdf:about=""
      <dc:format>image/svg+xml</dc:format>
      <dc:type
        rdf:resource="http://purl.org/dc/dcmitype/StillImage" />
      <dc:title></dc:title>
    </cc:Work>
  </rdf:RDF>
</metadata>
<g
  inkscape:label="rojo"
  inkscape:groupmode="layer"
  id="layer1">
  <rect
    style="opacity:1;fill:#ffaaaa;fill-opacity:1;stroke:#000000;stroke-opacity:1"
    id="rect4136"
    width="191.42857"
    height="240"
    x="34.285713"
    y="83.790779" />
</g>
<g
  inkscape:groupmode="layer"
  id="layer2"
  inkscape:label="verde">
  <ellipse
    style="opacity:1;fill:#aad400;fill-opacity:1;stroke:#000000;stroke-opacity:1"
    id="path4139"
    cx="411.42856"
    cy="266.64792"
    rx="111.42857"
    ry="108.57143" />
</g>
</svg>
```

El cual genera el siguiente proyecto:



De forma general la estructura que se crea al empezar un nuevo proyecto es un árbol donde el elemento raíz es del tipo o tag ‘SVG’ que sería el padre que contiene a todos los demás elementos.

Los elementos que cuelgan del nodo raíz por defecto son:

- *sodipodi:namedview*: que es el folio o superficie inicial que Inkscape crea en el proyecto (por defecto aparece con unos parámetros de tamaño, estilo, etc que son totalmente modificables).
- *svg:defs*: es un conjunto de perspectivas donde se configuran las vistas y tamaños (zooms) del proyecto.
- *svg:metadata*: subárbol encargado de almacenar el formato o lenguaje en el que se va a guardar el proyecto, en la mayoría de casos en xml.

A partir de entonces los nodos que nos podemos encontrar dentro de SVG son capas (que a su vez contendrán otras capas u objetos) u objetos que cuelguen directamente del nodo raíz (es decir, que no pertenezcan a ninguna capa).

Los objetos que podemos encontrar en un fichero SVG son:

- *svg* (elemento raíz).
- *namedview* (datos relacionados con la ventana de Inkscape y del proyecto).
- *defs* (grupo que engloba las perspectivas).
- *perspective* (encargados de las vistas del proyecto).
- *metadata* (conjunto de datos relacionados con la configuración del proyecto).
- *format* (formato en el que se almacenan los datos, en mi caso es en SVG + xml).
- *g* (puede ser una clase o un grupo de objetos como las cajas3d).
- *path* (elementos de dibujo ‘sin patrón’. Contienen muchos datos de coordenadas).
- *ellipse* (elipses y circunferencias).
- *rect* (rectángulos).
- *text* (elementos de texto).
- *line* (líneas).
- *polygon* (polígonos cerrados)

Los *paths* son unos de los *shapes* o elementos más utilizados ya que son los encargados de dibujar líneas “libres”. Dentro de su contenido podemos encontrar letras entre la multitud de coordenadas, cada una de ella significa lo siguiente:

- M = Mover hasta
- L = Línea hasta
- H = Línea horizontal hasta
- V = Línea vertical hasta
- C = Curva hasta
- S = Curva suave hasta
- Q = Curva de Bézier cuadrática hasta
- T = Curva de Bézier cuadrática suave hasta
- A = Arco elíptico
- Z = Cerrar circuito o *path*

---

#### 4.3. Archivos xml .inx

Los ficheros con extensión *.inx* son archivos implementados en XML encargados de mostrar la interfaz gráfica que ayuda a los usuarios a configurar los atributos.

A continuación se mostrará un ejemplo de este tipo de archivos describiendo que es cada una de las líneas que nos podemos encontrar en ellos.

```
<?xml version="1.0" encoding="UTF-8"?>
<inkscape-extension xmlns="http://www.inkscape.org/namespace/inkscape/
extension">
```

El comienzo de los archivos siempre será **la versión de XML** y la dirección de extensiones de inkscape.

```
<_name>Nombre</_name>
```

Nombre de la extensión. Será el que aparezca en el menú de inkscape.

```
<id>org.inkscape.color.custom</id>
```

Identificador único de la extensión.

```
<dependency type="executable" location="extensions">coloreffect.py</
dependency>
<dependency type="executable" location="extensions">color_custom.py</
dependency>
<dependency type="executable" location="extensions">simplestyle.py</
dependency>
```

Dependencias o paquetes que necesita nuestra extensión para su correcto funcionamiento.

En caso de que la ejecución necesite algún parámetro tendremos seguro una ventana en la “interfaz” de la extensión.

De manera adicional y opcional se puede añadir una pestaña extra de ayuda donde se comente cual es el funcionamiento de la extensión y que son cada uno de los parámetros, en este caso

tendremos que utilizar un parámetro (param) de tipo **notebook** con varios **page** para cada pestaña con su nombre y el texto que muestra

```
<param name="tab" type="notebook">
    <page name="Options" _gui-text="Options">
        .....
        </page>
    <page name="Help" _gui-text="Help">
        .....
        </page>
</param>
```

En este caso tenemos un notebook donde sus hojas son: Options (mostrando “Options” en la interfaz) y Help (mostrando “Help”).

Possibilidades dentro de las páginas:

- **Campos de enteros (int)** con título, mínimo, máximo, la etiqueta asociada que en este caso sería “Valor Numerico” y valor por defecto (en este caso es 1):

```
<param name="valor" type="int" min="1" max="6" _gui-text="Valor
Numerico">1</param>
```

- **Campos de texto (string)** con título, valor por defecto, que este caso es r, y la etiqueta asociada que es “Red Function”):

```
<param name="r" type="string" _gui-text="Red Function:">r</param>
```

- **Botones radio (optiongroup)** con un título para la sección colocado en \_gui-text, un nombre identificador y las múltiples opciones identificadas y con su contenido (la primera opción aparecerá marcada por defecto):

```
<param name="scale" type="optiongroup" _gui-text="Input (r,g,b)
Color Range:">
    <option value="1">0 - 1</option>
    <option value="255">0 - 255</option>
</param>
```

- **Checkboxes (boolean)** con un título, identificador e inicializados al valor que se le pase.

```
<param name='in-out-path' type="boolean" _gui-text="Create in-out
paths" >True</param>
```

- Desplegable (optiongroup appearance=“minimal”) que contará con un título, tipo y apariencia y todas las posibilidades internas, todas ellas identificadas y con un valor:

```
<param name="in-out-path-type" _gui-text="In-out path type:"
type="optiongroup" appearance="minimal">
    <_option value="Round">Round</_option>
    <_option value="Perpendicular">Perpendicular</
_option>
    <_option value="Tangent">Tangent</_option>
</param>
```

- Valores ocultos, en caso de que queramos mandar al fichero ejecutable python una variable pero no queramos mostrarla en la interfaz de la extensión tendríamos que usar el parámetro gui-hidden evaluado a true:

```

<param name="unit" type="string" gui-hidden="true">mm</param>

<page name="Help" _gui-text="Help">
    <_param name="instructions" type="description"
        xml:space="preserve">Todo lo que queramos escribir en la descripción
        de la página de ayuda</_param>
</page>
```

Dentro de la página de **ayuda** tendremos que crear un parámetro con un nombre y de tipo “description” (con el espacio XML preservado) donde incluiremos toda la documentación sobre el funcionamiento o utilización de la extensión que pueda servirle de ayuda al usuario.

```

<effect>
    <object-type>all</object-type>
    <effects-menu>
        <submenu _name="Color"/>
    </effects-menu>
</effect>
```

En estas líneas seleccionaremos todos los elementos de la extensión sobre los que hace efecto y diremos el nombre del submenu donde deberá ir colocada, en este caso es en color.

```

<script>
    <command reldir="extensions" interpreter="python">color_custom.py</
    command>
</script>
</inkscape-extension>
```

Dentro del apartado script crearemos una etiqueta *command* donde indicaremos cual va a ser el interprete que se necesitará para ejecutar nuestro fichero y cual es el nombre de este.

#### 4.4. Archivos python .py

Los archivos python, en mi caso, son los encargados de gestionar y modificar el documento SVG para realizar las modificaciones oportunas.

La estructura de estos fichero siempre será igual:

1. En la primera parte de los archivo .py nos encontramos un bloque de comentarios donde se indica:
  - El path donde se encuentra python.
  - El/los Copyright correspondientes en caso de que los tuviese la extensión.
  - Una breve descripción del funcionamiento de la extensión o temas relacionados con la posibilidad de distribución y/o modificación de la extensión en cuestión.
2. Importaciones necesarias para el correcto funcionamiento.
3. Cabecera de la clase donde indicaremos el nombre.
4. Definición del constructor y las funciones.

---

## 4.5. LXML (Element)

LXML<sup>[11]</sup> es una biblioteca estándar de python basada en XML que combina la velocidad de XML con la simplicidad de una API nativa de Python.

Los objetos de los proyectos de Inkscape son instancias de la clase Element de LXML.

Cada uno de estos objetos tiene unas propiedades o atributos que son:

- **attrib**: Formado por un diccionario con los atributos del elemento. Para conseguir o modificar dichos valores se podrá utilizar `.get()` `.set()` `.keys()` `.values()` y `.items()`.
- **tag**: Devuelve una URL seguido del tipo de objeto que es. Los tipos de tags con los que se suele trabajar son:

Los métodos que se pueden ejecutar sobre un objeto de la clase element son:

- `addnext(element)`: añade a element como hermano después del elemento que haya llamado al método.
- `addprevious(element)`: añade a element como hermano antes del elemento que haya llamado al método.
- `append(element)`: añade a element como subelemento del elemento que haya llamado al método.
- `clear()`: Resetea un elemento borrando todos los subelementos, esto es, borrando todos los atributos y poniendo todas las propiedades a None.
- `find(path, namespaces=None)`: devuelve el primer subelemento que coincida con el tag (con el formato completo, URL) o `path` pasado como argumento.
- `.findall(path, namespaces=None)`: devuelve todos los subelementos que coincidan con el tag (con el formato completo, URL) o `path` pasado como argumento.
- `set(clave, value)`: Modifica el valor de la clave por `value`.
- `get(key, default=None)`: devuelve un elemento de los atributos del objeto. Por ejemplo: `objeto.get('style')`.
- `getchildren()`: Devuelve todos los hijos directos del elemento que llama al método.
- `getiterator(tag=None, *tags)`: Devuelve una secuencia de todos los elementos en el árbol explorándolo en profundidad.
- `getnext()`: Devuelve el siguiente hermano o None
- `getparent()`: Devuelve el padre del nodo que llamó al método.
- `getprevious()`: Devuelve el hermano anterior o None.
- `getroottree()`: Devuelve un ElementTree con todos los subelementos del objeto que llamó al método donde este es la raíz.
- `index(child, start=None, stop=None)`: Encuentra la posición del hijo a partir del nodo que llamó al método.
- `insert(index, element)`: Inserta element en la posición index.
- `items()`: Devuelve los atributos del elemento en orden arbitrario.
- `iter(tag=None, *tags)`: Itera por todos los elementos del subárbol con orden de profundidad. Por ejemplo: `for nodo in svg.iter()`
- `iterancestors(tag=None, *tags)`: Itera por los padres del nodo que llamó al método.
- `iterchildren(tag=None, *tags)`: Itera por los hijos del nodo que llamó al método.
- `iterdescendants(tag=None, *tags)`: Itera por todos los descendientes del elemento en el orden en el que aparezcan en el documento.
- `iterfind(path, namespaces=None)`: Itera por los subelementos de todos los nodos que coincidan con el tag o `path`.

- `itersiblings(tag=None, preceding=False, * tags)`: Itera por los siguientes o anteriores hermanos del nodo que llamó al método.
- `keys()`: Devuelve, en orden arbitrario, una lista con los nombres de los atributos.
- `makeelement(_tag, attrib=None, nsmap=None, **_extra)`: Crea un nuevo elemento asociado con el mismo document.
- `remove(element)`: Elimina el elemento `element` (siempre que aparezca en el árbol por abajo, es decir, como descendiente del elemento que llamó al método).
- `replace(elementoViejo, elementoNuevo)`: Reemplaza el subelemento `elementoViejo` por el `elementoNuevo`.
- `set(key, value)`: Modificamos el valor del atributo `key` por el valor `value`.
- `values()`: Devuelve los valores de los atributos como una secuencia de `string` en orden arbitrario.

## 4.6. *Inkex.py*

Este módulo encapsula el comportamiento básico de una extensión permitiendo que el programador se centre en la manipulación de datos SVG. Además proporciona un método abstracto que tenemos que sobrecargar, `Effect()`.

Métodos de clase que nos proporciona:

- `effect()`: aquí se ejecutará el código que hayamos implementado para cada extensión.
- `affect()`: ejecuta el script.
- `debug(algo)`: muestra por pantalla el contenido del dato pasado como atributo, ya sea una variable o una cadena de texto.
- `localize()`: para la traducción de la extensión.
- `addNS(tag, ns=None)`: Se utiliza para devolver el valor de “ns:tag” del elemento o para modificarlo.
- `errmsg_(‘Descripción’)`: Devuelve el error con la descripción indicada por pantalla.

Métodos sobre instancias de la clase:

- `parse(self, filename=None)`: Lee un fichero que se le pase como argumento y lo intenta devolver de manera que sea entendible.
- `getselected()`: devuelve una lista con los objetos seleccionados en Inkscape.
- `getElementById(id)`: devuelve el elemento identificado por el id pasado como argumento.
- `getParentNode(node)`: devuelve el nodo padre del nodo pasado como argumento.
- `getNamedView()`: devuelve el objeto encargado de la vista del proyecto.
- `createGuide(x, y, g)`: crea una guía en Inkscape pasando por x e y con g grados de inclinación.
- `uniqueId(idViejo, True)`: Devuelve un ID único añadiendo dígitos al idViejo pasado como argumento.
- `getDocumentUnit()`: Devuelve las unidades utilizadas en el proyecto.

Atributos:

- `documents`: DOM (Modelo de objetos del documento).
- `selected`: lista con los nodos seleccionados in Inkscape.
- `doc_ids`: diccionario con todos los identificadores utilizados en el documento y el número de usos de cada uno.
- `options`: parámetros pasados al script.

---

## 4.7. Simplestyle.py

Este módulo nos proporciona métodos para tratar los datos de diseño dentro del fichero SVG:

- `parseStyle(string)` : crea un diccionario con los valores del style de un objeto. Donde las claves serán los enunciados y el contenido será el atributo.
- `formatStyle(dict)` : función inversa a la anterior. Crea una línea de estilo a partir del diccionario pasado como parámetro.
- `isColor(color)` : Devuelve true o false dependiendo si el parámetro pasado tiene formato de color o no (formato color = #123456 o #123).
- `parseColor(color)` : Pasándole como atributo un color en formato hexadecimal devuelve un array de enteros (r, g, b) representando un color RGB.
- `formatColoria(array)` : A partir de un array de int obtiene un #rrggb
- `formatColorfa(array)` : A partir de un array de float obtiene un #rrggb
- `formatColor3i(r, g, b)` : A partir de tres enteros obtiene #rrggb
- `formatColor3f(r, g, b)` : A partir de tres floats obtiene #rrggb
- `svgcolors`: diccionario donde se tienen los nombres de colores y su valor en hexadecimal.

---

## 4.8. Línea de comandos

Al instalar Inkscape estamos instalando un programa con interfaz gráfica a la vez que instalamos una versión del mismo programa para ejecutar desde la línea de comandos.

En Windows una vez que instalamos la aplicación se creará el enlace con la consola de comandos para que se pueda ejecutar Inkscape directamente desde la terminal pero en Linux y Mac OS tendremos que crear un acceso directo desde la carpeta `/usr/local/bin/` al archivo de Inkscape que se suele encontrar en la siguiente ruta: `/Applications/Inkscape.app/Contents/Resources/bin/inkscape` (Path extraído de Mac OS).

Una vez que tengamos configurada la línea de comandos para ejecutar Inkscape podremos hacer ciertas tareas sobre archivos SVG desde la línea de comandos, estas tareas son:

**-?, --help**

Muestra un mensaje de ayuda

**-V, --version**

Muestra la versión de Inkscape

**-a x0:y0:x1:y1, --export-area=x0:y0:x1:y1**

Cuando exportamos en SVG, configuramos el área del documento SVG que se va a exportar desde el documento SVG (utilizando la unidad establecida por Inkscape). Por defecto exporta el documento entero. El punto (0,0) será el punto izquierdo de abajo.

**-C, --export-area-page**

Cuando exportamos PNG, PDF, PS, y EPS, se exporta “la página” del documento SVG (“el tablero de dibujo”). En las exportaciones a PNG, PDF y PS esta es la opción por defecto. Para exportaciones a EPS no es la opción por defecto porque se exporta la caja contenedora de los objetos a exportar y en caso de que sobrepasen el tablero se exporta hasta los límites de la página.

**-D, --export-area-drawing**

En las exportaciones a PNG, PDF, PS y EPS se exporta el área de dibujada no la página.

**-b COLOR, --export-background=COLOR**

Se utiliza para realizar exportaciones a PNG configurando el color de fondo del archivo resultante. El formato de COLOR es "#XXXXXX", en hexadecimal.

**-d DPI, --export-dpi=DPI**

Resolución utilizada para exportaciones a PNG, PS, EPS o PDF. El valor por defecto es 96dpi.

**-e FILENAME, --export-png=FILENAME**

Especifica el nombre del fichero donde se guardará la exportación en png. Si ya existe lo sobrescribirá sin preguntar.

**-f FILENAME, --file=FILENAME**

Abre el documento indicado, también se puede omitir la opción "-f".

**-g, --with-gui**

Intenta utilizar la interfaz gráfica.

**-z, --without-gui**

No utiliza la interfaz gráfico para la ejecución del comando.

**-h HEIGHT, --export-height=HEIGHT**

Configura la altura del mapa de pixeles, en píxeles.

**-i /ID, --export-id=/ID**

Con esta variable indicamos el ID de los objetos que queremos exportar, todos los demás no se incluirán en el documento exportado.

**-l, --export-plain-svg=FILENAME**

Exporta documentos al formato svg simple, sin *sodipodi*: ni *inkscape: namespaces* ni ningún metadato RDF.

**-x, --extension-directory**

Lista el directorio que Inkscape tiene configurado como directorio de extensiones. Esto se usa para extensiones externas.

**--verb-list**

Muestra la lista de verbos que se pueden utilizar para configurar la ejecución de Inkscape desde la línea de comandos.

**--verb=VERB-ID, --select=OBJECT-ID**

--verb ejecutará el verbo indicado como si se pulsase desde un botón o un menú.

--select hace que se ejecute la opción del verbo señalado sobre los objetos seleccionados.

**-w WIDTH, --export-width=WIDTH**

Configura la anchura del mapa de bits generado, en pixeles.

**-P FILENAME, --export-ps=FILENAME**

Especifica el nombre del fichero donde se guardará la exportación en PS. Si ya existe lo sobrescribirá sin preguntar.

**-E FILENAME, --export-eps=FILENAME**

Especifica el nombre del fichero donde se guardará la exportación en EPS. Si ya existe lo sobrescribirá sin preguntar.

**-A FILENAME, --export-pdf=FILENAME**

Especifica el nombre del fichero donde se guardará la exportación en PDF. Si ya existe lo sobrescribirá sin preguntar.

**--export-pdf-version=PDF-VERSION**

Selecciona la versión de PDF que se le indique.

**-T, --export-text-to-path**

Convert text objects to paths on export, where applicable (for PS, EPS, PDF and SVG export).

#### **--export-ignore-filters**

Para exportaciones a PS, EPS y PDF se ignorarán los filtros como el difuminado, por ejemplo.

#### **-I, --query-id**

Set the ID of the object whose dimensions are queried. If not set, query options will return the dimensions of the drawing (i.e. all document objects), not the page or *viewbox*

#### **-X, --query-x**

Query the X coordinate of the drawing or, if specified, of the object with --query-id. The returned value is in px (SVG user units).

#### **-Y, --query-y**

Query the Y coordinate of the drawing or, if specified, of the object with --query-id. The returned value is in px (SVG user units).

#### **-W, --query-width**

Query the width of the drawing or, if specified, of the object with --query-id. The returned value is in px (SVG user units).

#### **-H, --query-height**

Query the height of the drawing or, if specified, of the object with --query-id. The returned value is in px (SVG user units).

#### **-S, --query-all**

Prints a comma delimited listing of all objects in the SVG document with IDs defined, along with their x, y, width, and height values.

#### **--shell**

Con este parámetro, Inkscape entrará en un modo de línea de comandos interactivo. En este modo, se irán escribiendo los comandos e Inkscape los irá ejecutando sin tener que escribir “inkscape” para cada nuevo comando. Esta opción es útil generalmente para servidores donde se mandarán bastantes operaciones por línea de comandos.

#### **--g-fatal-warnings**

Esta opción fuerza a Inkscape a cancelar la ejecución de cualquier comando en caso de que aparezca algún *warning*. Suele resultar útil para depuración.

#### **--usage**

Muestra un breve mensaje con instrucciones de uso.

#### **-I, --query-id**

Identifica al objeto sobre el que vamos a preguntar algún dato.

#### **-X, --query-x**

Devuelve la coordenada X del dibujo o si está especificado con “—query-id” del objeto identificado. Devuelve el valor en píxeles que es la unidad por defecto de Inkscape.

#### **-Y, --query-y**

Devuelve la coordenada Y del dibujo o si está especificado con “—query-id” del objeto identificado. Devuelve el valor en píxeles que es la unidad por defecto de Inkscape.

#### **-W, --query-width**

Devuelve la anchura del dibujo o si está especificado con “—query-id” del objeto identificado. Devuelve el valor en píxeles que es la unidad por defecto de Inkscape.

#### **-H, --query-height**

Devuelve la altura del dibujo o si está especificado con “—query-id” del objeto identificado. Devuelve el valor en píxeles que es la unidad por defecto de Inkscape.

#### **-S, --query-all**

Devuelve separados por comas todos los objetos en el documento SVG con sus ID, coordenada x, coordenada y, anchura y altura.

Algunos ejemplos de uso de la línea de comandos:

- Abrir un archivo en la interfaz gráfica:  
*inkscape filename.svg*
- Exportar un archivo SVG a PNG con la resolución por defecto 96dpi:  
*inkscape filename.svg --export-png=filename.png*
- Mismo ejemplo que el anterior pero forzamos a que el archivo tenga una resolución de 600x400 píxeles:  
*inkscape filename.svg --export-png=filename.png -w600 -h400*
- Exportamos solo el espacio dibujado, no toda la página:  
*inkscape filename.svg --export-png=filename.png --export-area-drawing*
- Convierte un documento SVG de Inkscape a un SVG simple:  
*inkscape filename1.svg --export-plain-svg=filename2.svg*
- Convierte un documento SVG a EPS pasando todos los textos a *paths*:  
*inkscape filename.svg --export-eps=filename.eps --export-text-to-path*
- Pregunta la anchura del objeto con identificador *text1555*:  
*inkscape filename.svg --query-width --query-id text1555*

---

## 4.9. Instalación de extensiones

Para instalar una extensión o un conjunto de estas tendremos que localizar los ficheros con extensiones *.inx* y *.py*, en caso de que se encuentre comprimidos habrá que descomprimirllos.

Habrá que colocarlos en el directorio donde Inkscape busca las extensiones para lanzarlas en su interfaz, que depende del sistema operativo se encuentra en un *path* u otro:

- Mac: *~.config/inkscape/extensions/*
- Linux: *~/.config/inkscape/extensions/*
- Windows: *C:\Program Files\Inkscape\share\extensions*

---

## 4.10. Ejemplo

En este apartado, para terminar la sección de guía, se va a exponer una extensión sencilla con ejemplo. Su función consiste en crear un cuadrado a partir de una coordenada y una longitud de lado.

Para ello se mostrará un archivo XML llamado *crearCuadrado.ink* donde se configura la coordenada de comienzo y la longitud del lado y un archivo python (*crearCuadrado.py*) que será el encargado de coger esos parámetros y crear el cuadrado a partir de ellos.

*crearCuadrado.ink*

```
<?xml version="1.0" encoding="UTF-8"?>
<inkscape-extension xmlns="http://www.inkscape.org/namespace/inkscape/
extension">
  <_name>Crear cuadrado</_name>
```

```

<id>plhurtado.crearCuadrado</id>
  <dependency type="executable" location="extensions">crearCuadrado.py</dependency>
  <dependency type="executable" location="extensions">inkex.py</dependency>
  <param name="tab" type="notebook">
    <page name="options" _gui-text="Options">
      <param name="x" type="int" default="0" _gui-text="Coordenada x del punto de comienzo: ">0</param>
      <param name="y" type="int" default="0" _gui-text="Coordenada y del punto de comienzo: ">0</param>
      <param name="longitud" type="float" default="20" min="1" max="500" _gui-text="Longitud de los lados: ">20</param>
    </page>
    <page name="help" _gui-text="Help">
      <param name="helpOptions" type="description">
        Esta extensión creará un cuadrado a partir de:
        -X: Valor x para la coordenada de comience.
        -Y: Valor y para la coordenada de comience.
        -Longitud: Longitud que tendrá cada uno de los lados del cuadrado.
      </param>
    </page>
  </param>
</effect>
<object-type>all</object-type>
<effects-menu>
  <submenu _name="TFG"/>
</effects-menu>
</effect>
<script>
  <command reldir="extensions" interpreter="python">crearCuadrado.py</command>
</script>
</inkscape-extension>

```

### crearCuadrado.py (descrito en los comentarios)

```

#!/usr/bin/env python
# -*- coding: utf-8 -*-

import sys, inkex, simplestyle
sys.path.append('/usr/share/inkscape/extensions')

class crearCuadrado(inkex.Effect):

    def __init__(self):

        # Recuperación de los parámetros.
        inkex.Effect.__init__(self)
        self.OptionParser.add_option("--x", action="store", type="int",
dest="x")
        self.OptionParser.add_option("--y", action="store", type="int",
dest="y")

```

```

        self.OptionParser.add_option("--longitud", action="store", type="float",
dest="longitud")
        self.OptionParser.add_option("--tab", action="store", type="string",
dest="tab")

def effect(self):
    #Guardo los parámetros en variables sencillas
    x = self.options.x
    y = self.options.y
    longitud = self.options.longitud

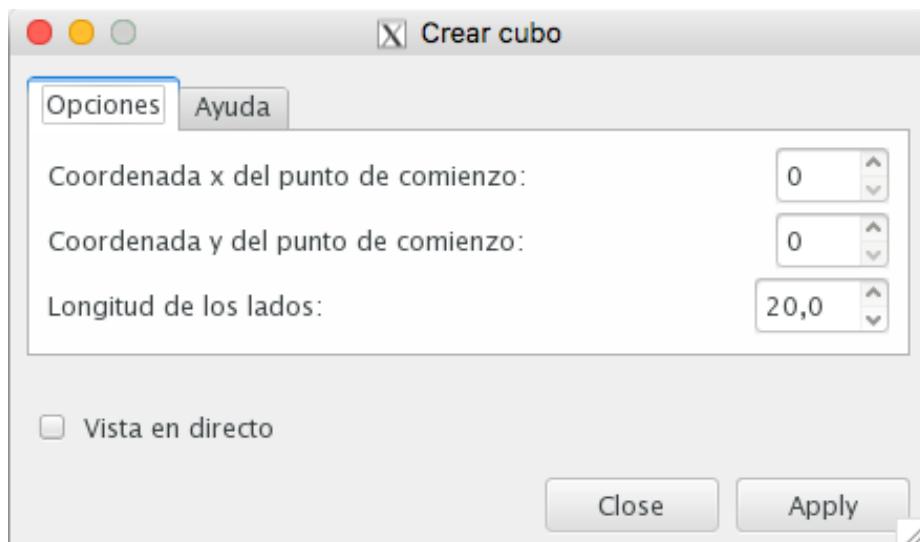
    #Configuración de coordenadas
    linea = 'M ' + str(x) + ',' + str(y) + ' L ' + str(x + longitud) + ',' +
str(y) + ' L ' + str(x + longitud) + ',' + str(y + longitud) + ' L ' + str(x) +
',' + str(y + longitud) + ' Z '

    #Estilo de la línea
    style={'stroke': '#FF0000', 'stroke-width': '1', 'fill': 'none'}
    #Estilo aplicado, nombre del objeto y coordenadas
    drw = {'style': simplestyle.formatStyle(style), inkex.addNS('label',
'inkscape'):'objetoCuadrado', 'd':linea}
    #Generamos el objeto como hijo de la capa actual
    inkex.etree.SubElement(self.current_layer, inkex.addNS('path', 'svg'),
drw)

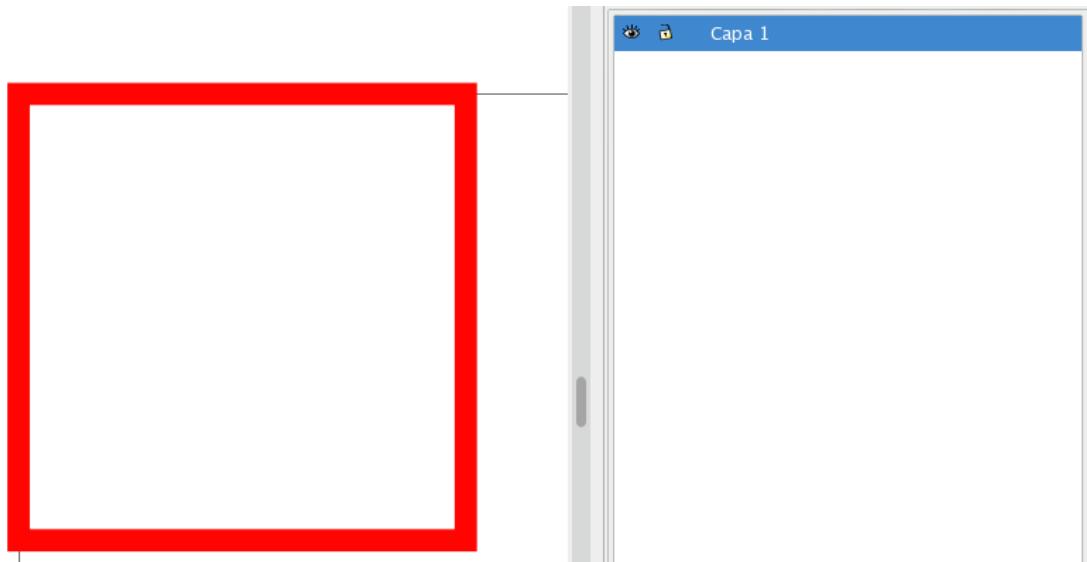
# Crea una instancia de effect y la aplica.
effect = crearCuadrado()
effect.affect()

```

Al ejecutar la extensión podemos ver la siguiente interfaz:



Y si mandamos los parámetros anteriores al ejecutable obtenemos el siguiente resultado:





## 5. EXTENSIONES

En este apartado se describirán en profundidad cada una de las extensiones que se han realizado o investigado a lo largo del desarrollo del proyecto.

Para cada una de ellas se crearán 5 apartados diferentes donde se expondrá

- La solicitud del cliente o tutor y por qué puede ser útil esta extensión para cualquier persona que trabaje con cortadoras láser.
- Análisis de requisitos funcionales y no funcionales
- Diseño de la extensión
- Implementación, donde se citarán las partes más interesantes del código.
- Y por último una batería de pruebas donde se han seleccionado alguna de los test más interesantes de entre todos los realizados.

---

### 5.1. Capa por color

El cliente nos comenta que habitualmente utiliza diferentes colores para los bordes de las figura que genera simbolizando así los diferentes tipos o diferentes intensidades de corte que necesitará cada figura.

Por esto, nos solicitado la implementación de una extensión que cree tantas capas como colores diferentes (de bordes) haya en el proyecto y que cada objeto se coloque dentro de la capa correspondiente.

Como indicación nos ha dicho que se intente mantener los grupos de objetos con el mismo color de borde, es decir, que los objetos que estén agrupados y tengan el mismo color de borde no se desagrupen.

#### A. Análisis

Requisitos funcionales:

RF1. Se creará una nueva capa por cada color diferente que aparezca en los bordes de los objetos del proyecto.

RF2. Se eliminarán las capas antiguas que hubiese en el proyecto.

RF3. Se tendrán en cuenta todos los objetos, tanto los que están dentro de alguna capa como los que cuelguen directamente del nodo raíz.

RF4. Los objetos compuestos, como son las cajas3d, se guardarán en una capa especial dejando intacto su agrupación para que sea más fácil su posterior manejo.

RF5. Los grupos de objetos donde todos tengan el mismo color en el borde no se disgregarán, se guardarán en su capa correspondiente sin separar.

RF6. A las capas se les pondrá como nombre un valor que sea intuitivo, por ejemplo, para la capa con los objetos de color #FFFFFF en el borde se le dará como nombre “black”. En caso de que no haya ningún color con nombre propio se nombrará con el valor en hexadecimal.

Requisitos no funcionales:

RNF1. Deberá implementarse en python.

RNF2. No habrá que pasarle parámetros para la ejecución. Se ejecutará directamente.

## B. Diseño

El proceso consistirá en recorrer todos los objetos y tendremos en cuenta varias cuestiones:

- Con las partes de las cajas3d no haremos nada.
- Con los grupos que engloban los objetos de las cajas3d crearemos una capa y si ya está creada añadiremos un nuevo elemento o grupo.
- Las capas las añadiremos a la lista de capas originales para eliminarlas al final.
- Para todos los demás nodos que tengan estilo (los objetos visibles del proyecto) analizaremos su color, miraremos si ya hay una capa para ese color y lo añadiremos y en caso de que no exista dicha capa la crearemos y añadiremos el nodo.

Por ultimo, eliminamos todas las capas originales, creamos todas las capas nuevas y colgamos los nodos de sus capas correspondientes.

## C. Implementación

Para el nombre de las capas he intentado escoger aquel que sea el más próximo al color del objeto.

Esto lo he conseguido calculando la distancia entre el color de cada una de las piezas del proyecto y todos los colores guardados en el diccionario del archivo `symplesyle.py` y escogiendo el que se encuentre a menor distancia (calculado igual que la separación entre dos coordenadas cartesianas en tres dimensiones ya que los colores tienen una estructura similar, R G B).

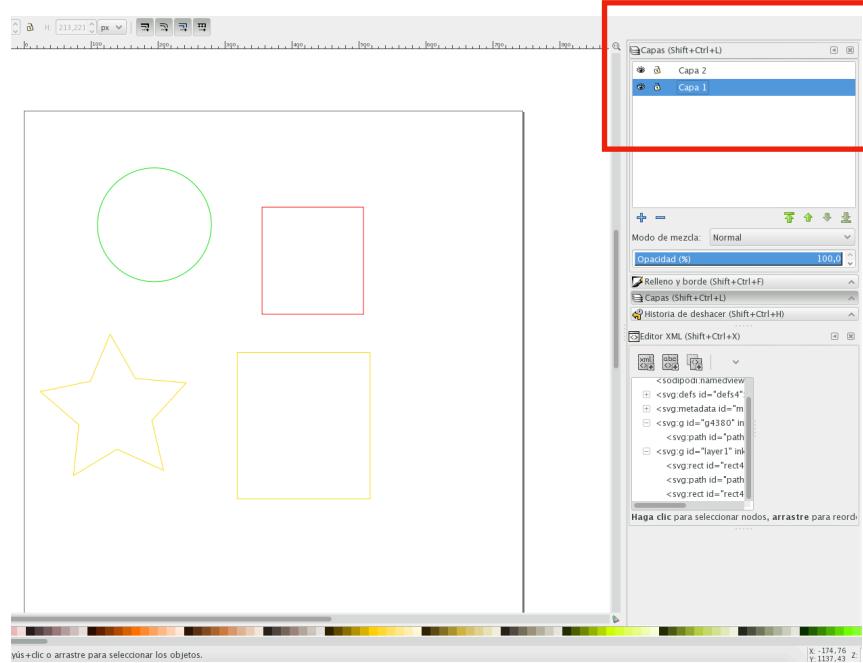
## D. Pruebas

**Nombre:** Prueba 1-1.

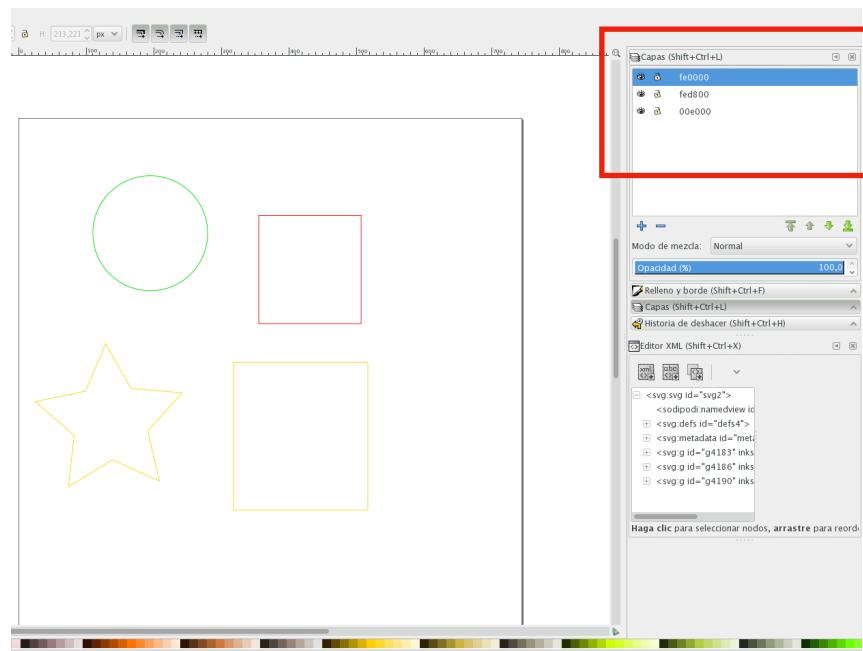
**Descripción:** Comprobar que se agrupan los diferentes objetos en capas según su color de borde.

**Resultado:** SATISFACTORIO.

**Estado inicial:**



**Estado final :**

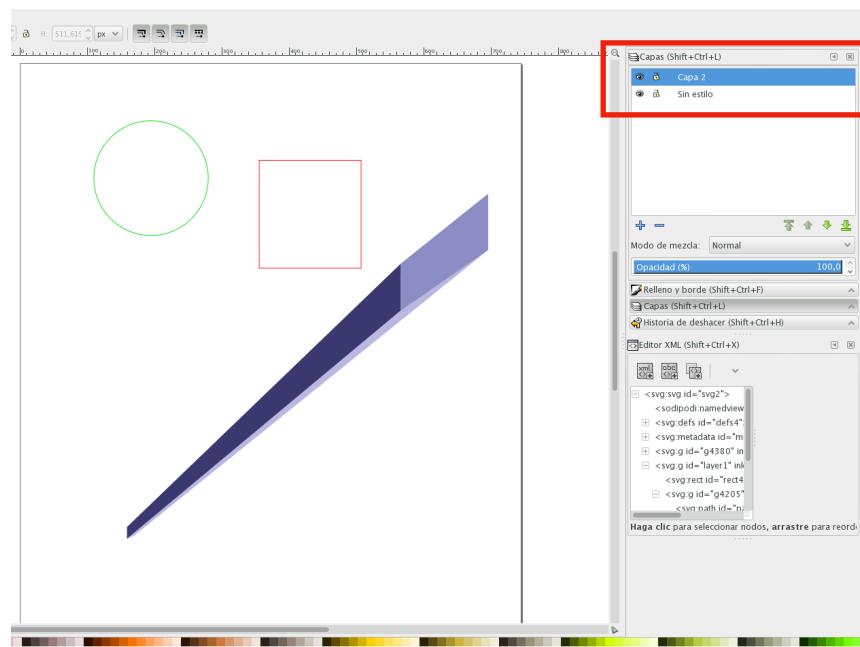


**Nombre:** Prueba 1-2.

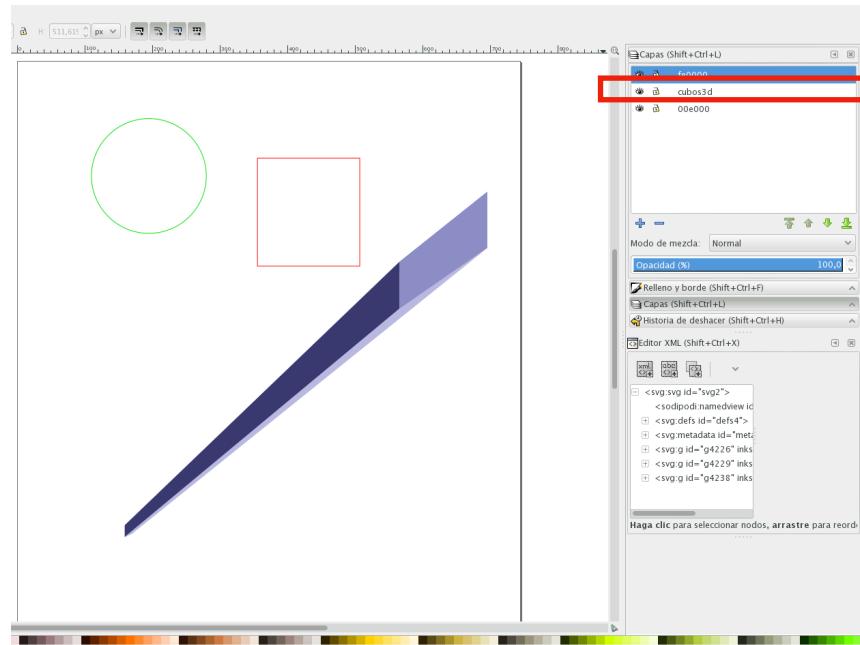
**Descripción:** Comprobar que se cree una capa especial para las cajas3d.

**Resultado:** SATISFACTORIO.

**Estado inicial:**



**Estado final:**



**Nombre:** Prueba 1-3.

**Descripción:** Comprobar que los nombres de las capas son descriptivos. Se le asigna un nombre relacionado con el color que se utiliza.

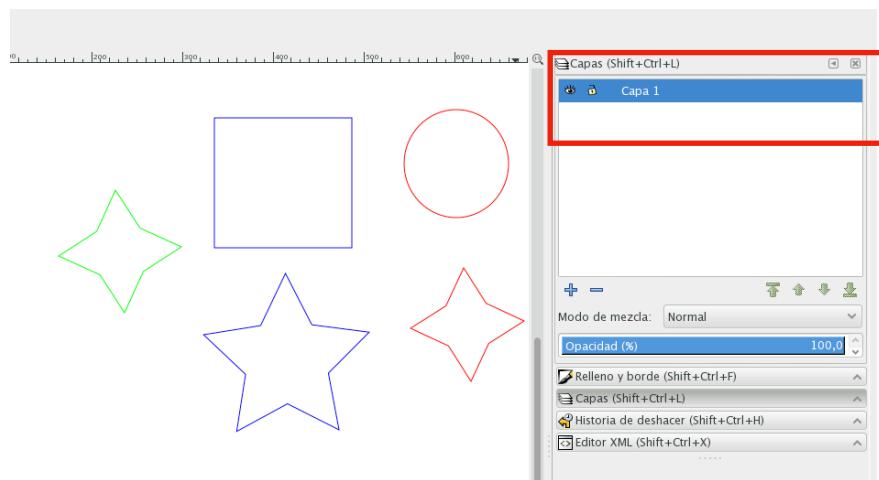
**Resultado:** FALLIDO.

**Descripción del fallo:** El número del color del objeto a estudiar se pasaba a formato RGB sin el carácter “#” por lo que el cálculo de sus tres valores era erróneo y ponía nombres de colores que no estaban cercanos al del objeto.

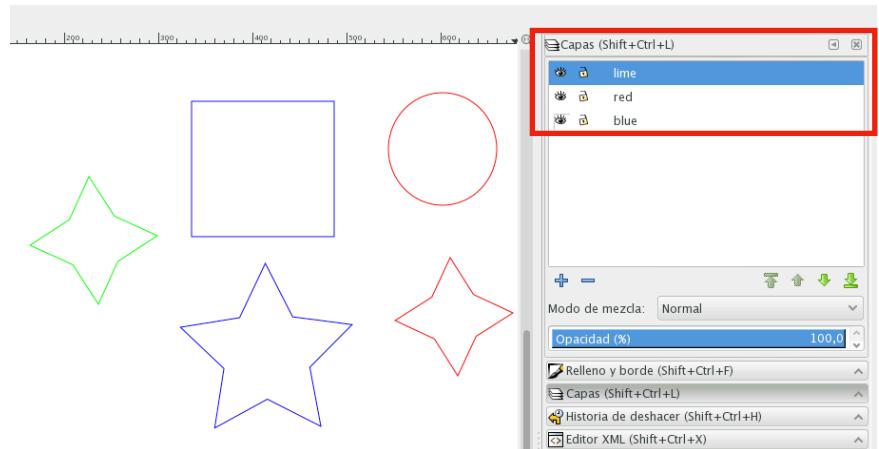
**Solución:** Como solución se le añade “#” al valor del color y después se convierten los valores hexadecimales a tres valores (R G B) y se calcula la distancia entre el color del objeto y cada uno de los colores declarados en el diccionario de colores estándar. Finalmente se escoge el color que esté a menor distancia.

El cálculo de la distancia se hace como el cálculo de la distancia entre dos puntos de tres coordenadas cada uno.

**Estado inicial:**



**Estado final:**



---

## 5.2. Color por capa

El cliente nos explica que habitualmente suele dibujar las figuras agrupándolas en capas diferentes de forma que cada una de ellas simbolice un corte distinto.

Por lo tanto, solicita una extensión que asigne el mismo color de borde a todos los objetos de una misma capa y que el color sea diferente entre las distintas capas para poder diferenciar de forma rápida y visual todos los objetos de un mismo tipo de corte.

La selección del color será aleatoria a no ser que el nombre de la capa sea un nombre de color ya definido, en ese caso se tendrá que poner dicho color como color de borde a los objetos de la capa.

Como extra también ha solicitado que se muestre un “checkbox” en la interfaz de la extensión para dar la opción al usuario de eliminar todas las capas y que los objetos (una vez cambiados de color) cuelguen del nodo raíz.

### A. Análisis

#### Requisitos funcionales:

RF1. Se le asignará un color a todos los objetos pertenecientes a alguna de las capas existentes en el proyecto.

RF2. El color será único entre capas. No podrá haber dos capas con el mismo color.

RF3. A los objetos que no estén dentro de ninguna capa no se les cambiará el color.

RF4. Si una capa tiene como etiqueta el nombre de un color definido se le asignará dicho color a sus hijos u objetos (siempre y cuando sea un color registrado en el diccionario de colores del archivo `symplesyle.py`).

RF5. La selección de color se hará de manera aleatoria.

RF6. El nombre de las capas del proyecto cambiará en caso de que el nombre no sea un color, en este caso se le denominará con el nombre anterior+colorUtilizado.

RF7. En caso de que se pulse el “checkbox” se eliminarán todas las capas después de aplicarle el color a los objetos dejando a estos colgando del nodo raíz.

#### Requisitos no funcionales:

RNF1. Deberá implementarse en python.

### B. Diseño

#### Parámetros en la interfaz:

Se creará un *checkbox* que se pueda marcar en caso de querer que se eliminen todas las capas después de haberle aplicado el cambio de color según las capas donde se encuentrasen.

Algoritmo:

El proceso consistirá en recorrer todas las capas del proyecto y para cada una de ellas:

- Comprobamos si el nombre de la capa es un nombre de color definido para asignarlo como color de todos sus objetos y si no se escoge uno de manera aleatoria, en ambos casos comprobando que no se repitan colores.
- Recorremos todos los hijos que cuelgan de la capa asignándole el color seleccionado.

### C. Implementación

Se han implementado dos métodos externos:

- cambioColor (estilo, color): Método encargado de modificar el color de un *estilo* pasado como parámetro por un *color* nuevo.
- colorAleatorio(): Método que genera un color en hexadecimal a partir de tres valores aleatorios (*r*, *g*, *b*)

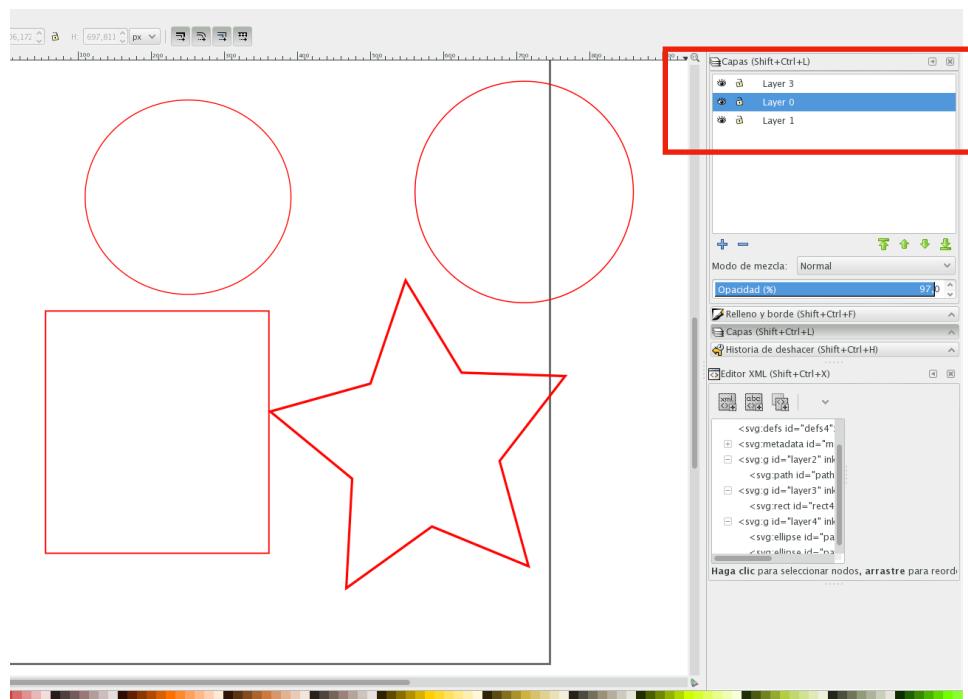
## D. Pruebas

**Nombre:** Prueba 2-1.

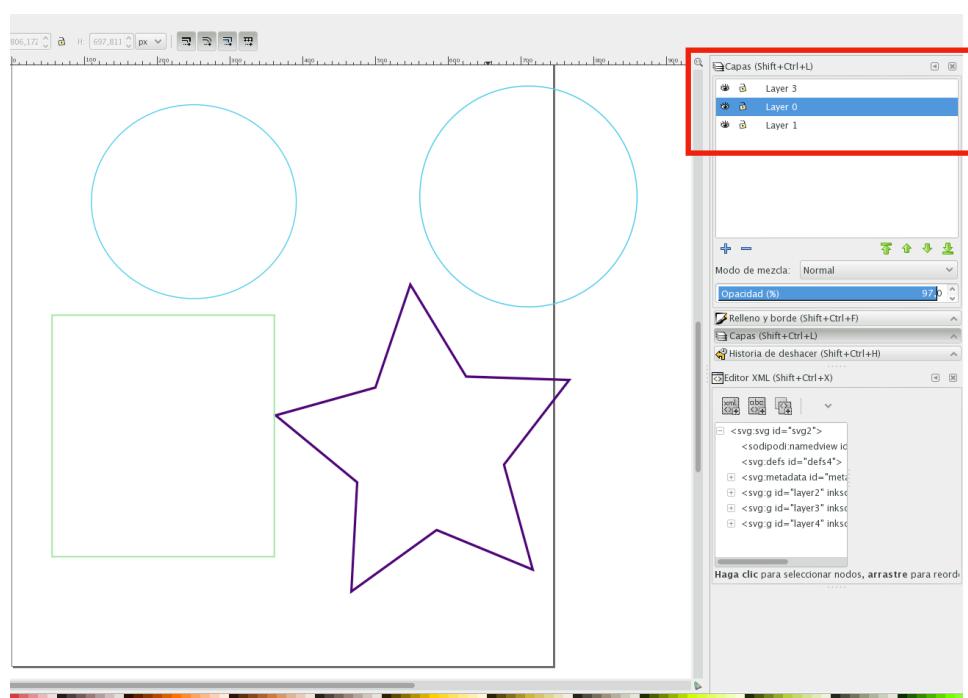
**Descripción:** Comprobar que se le asigne un color diferente a cada una de las capas existentes y que el color de borde de cada uno de los objetos cambia de manera correcta.

**Resultado:** SATISFACTORIO.

**Estado inicial:**



**Estado final:**

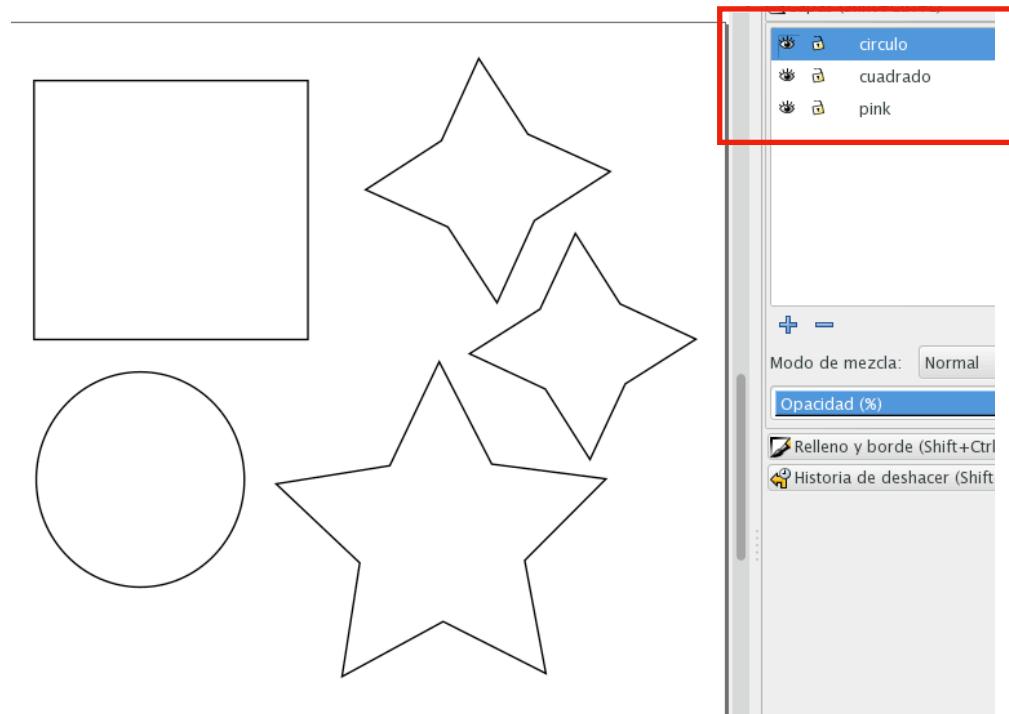


**Nombre:** Prueba 2-2.

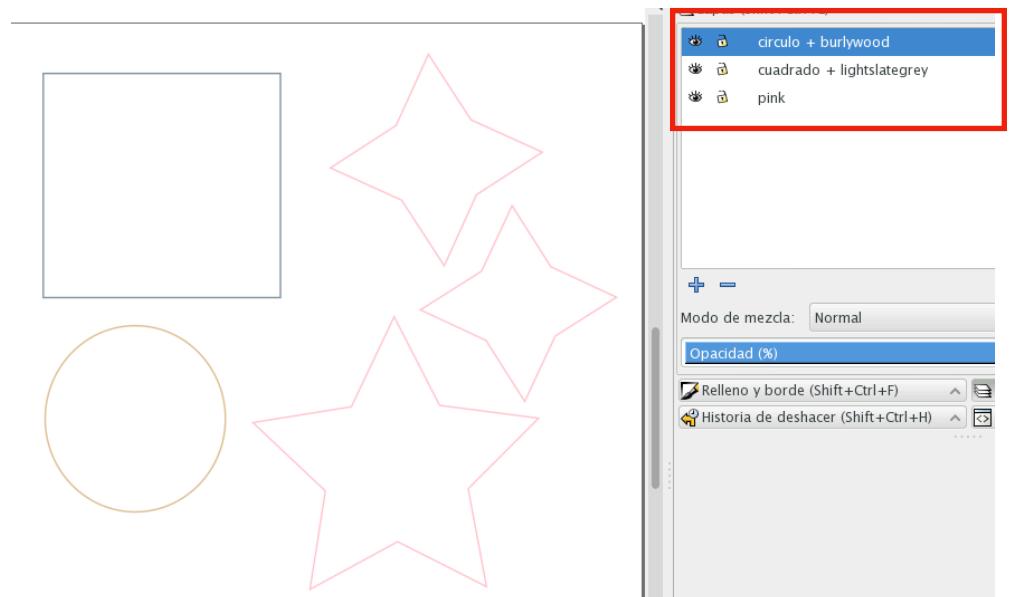
**Descripción:** Comprobar que si el nombre de una capa es un color definido se le asigne dicho color a todos los objetos de esta capa.

**Resultado:** SATISFACTORIO.

**Estado inicial:**



**Estado final:**



**Nombre:** Prueba 2-3.

**Descripción:** Comprobar que al aplicarle un color a una capa cuyo nombre no fuese el de un color definido el nombre original de la capa se mantiene y se añade el nombre del color aplicado a las figuras.

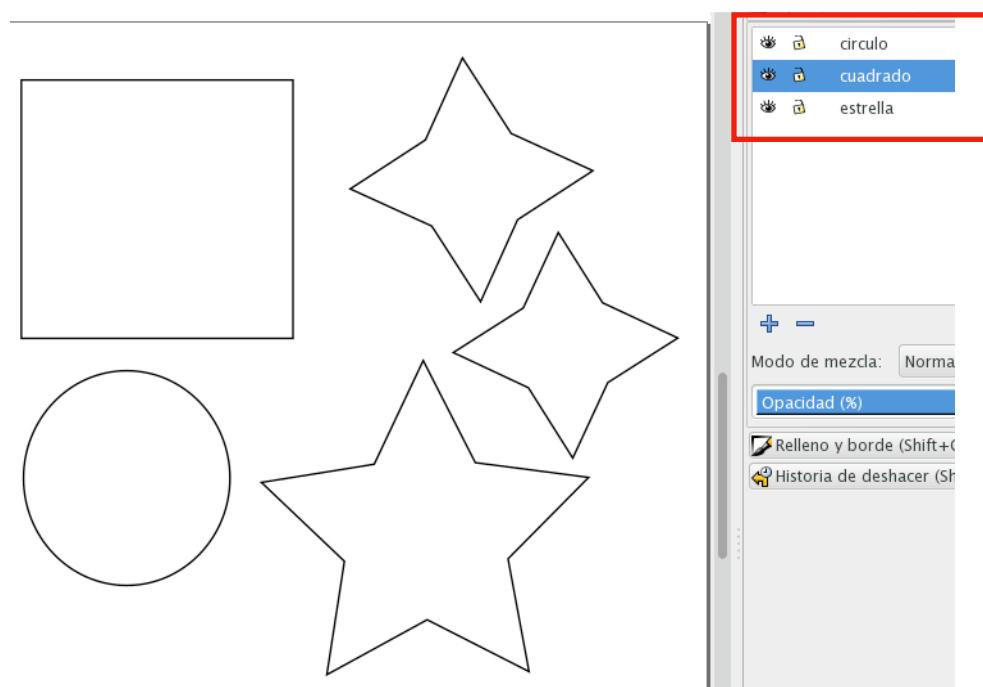
**Resultado:** FALLIDO.

**Descripción del problema:** Al ejecutar varias veces la misma extensión los colores aplicados se iban acumulando en el nombre de las capas.

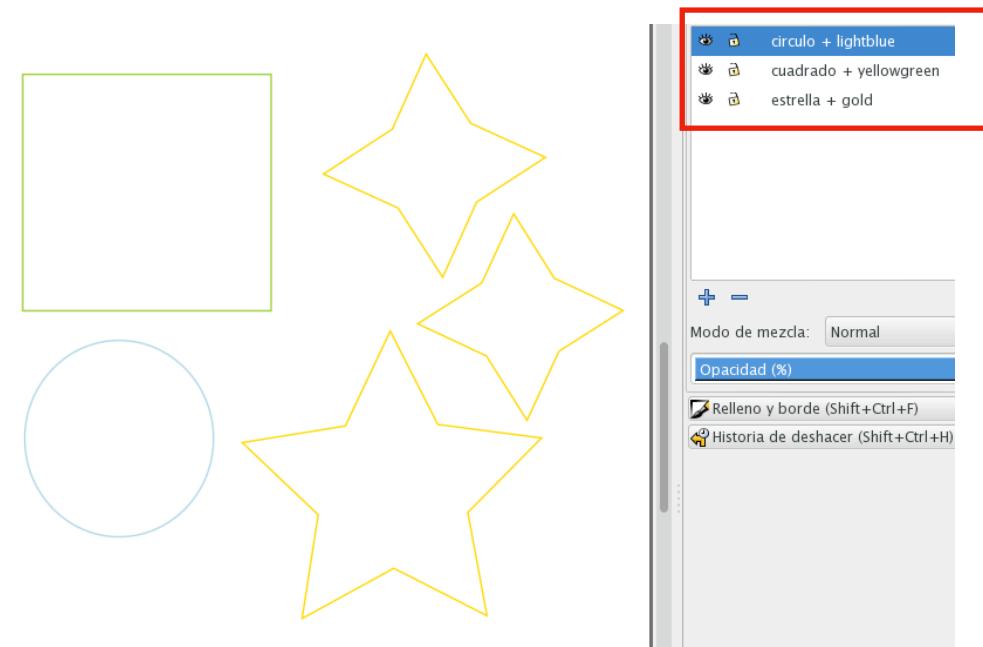
**Solución:** Para solucionar este problema he tenido en cuenta que se pueda ejecutar la extensión varias veces para cambiar el color de las capas. En este caso, tan solo se le cambia el nombre del color manteniendo el nombre original de la capa.

Para ello he tenido que buscar el elemento “+” que es el que une el nombre original con el color aplicado y cambiar el color a partir de ese punto.

**Estado inicial:**



**Estado final:**

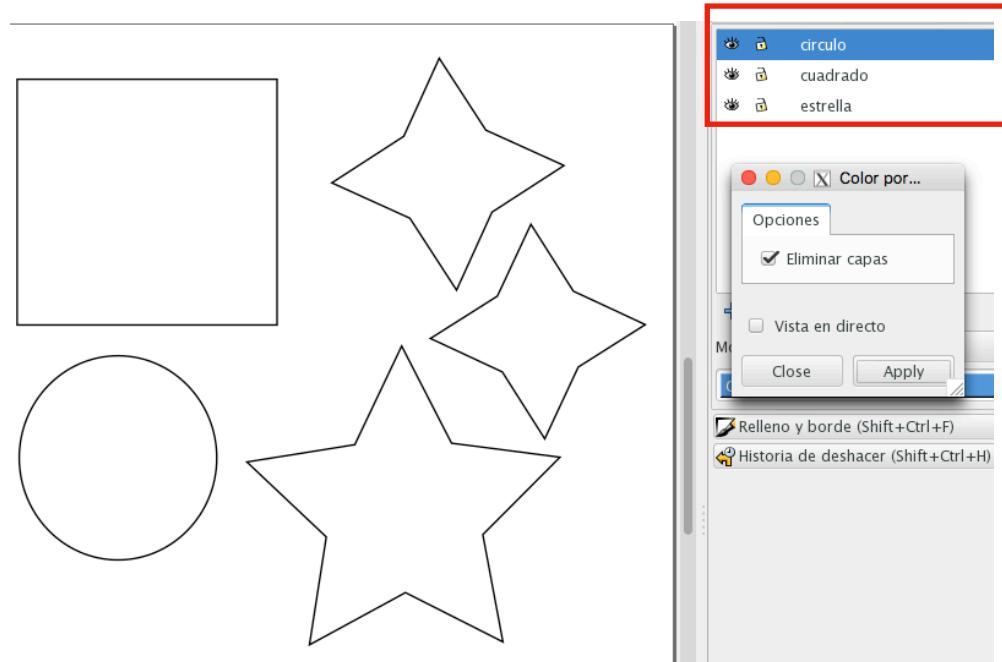


**Nombre:** Prueba 2-4.

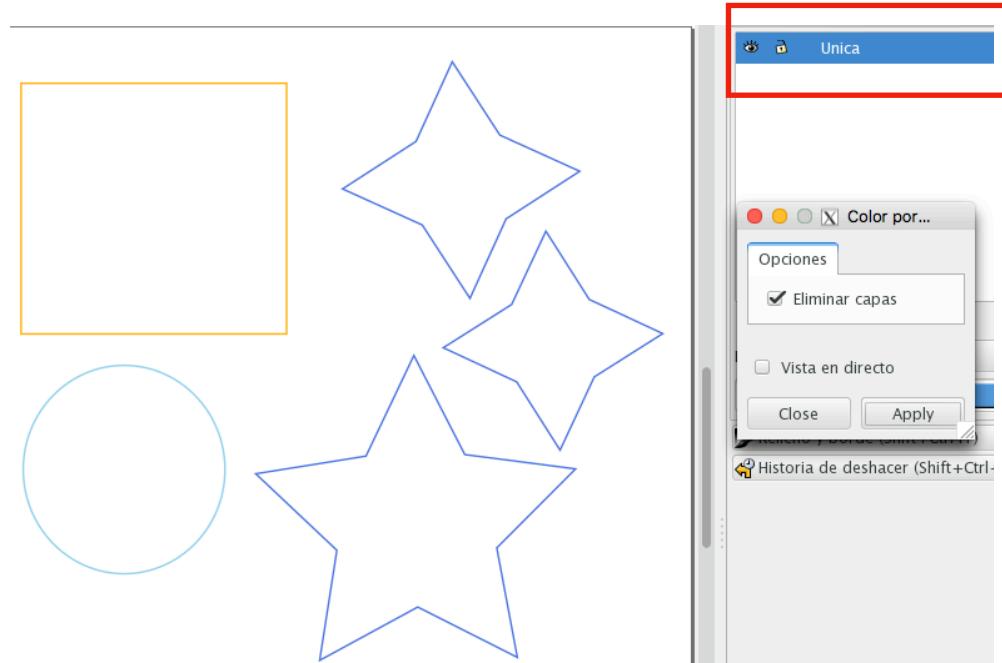
**Descripción:** Comprobar la funcionalidad del botón “Eliminar capas”. Se deberán aplicar los colores a los objetos de las diferentes capas y tras esto eliminar todas las capas y dejarlos colgando de una capa única.

**Resultado:** SATISFACTORIO.

**Estado inicial:**



**Estado final:**



---

### 5.3. TabbedBoxMaker

El cliente solicita que a partir de una extensión ya creada [TabbedBoxMaker] se genere una nueva donde se añada una acción que dé la opción de que todas las líneas de un mismo *shape* se almacenen en un único *Path*.

Además nos indica que su cortadora láser hace algo extraño en las esquinas debido a que se duplican las coordenadas, tendré que investigar como solucionar dicho problema.

La aplicación que voy a modificar es de Paul Hutchison aunque esta a su vez es una adaptación o actualización de la versión original de Elliot White.

#### A. Análisis

##### Análisis de la extensión:

Esta extensión nos da la posibilidad de crear una caja totalmente configurable. Podremos modificar el número de lados dentados o crear separadores internos entre otras muchas opciones.

También ofrece la posibilidad de crear una caja abierta por uno o varios de sus lados.

Los parámetros configurables son:

- Unidad de medida utilizada.
- Si las medidas de la caja serán del interior o del exterior.
- Largura.
- Anchura.
- Altura.
- Anchura mínima de las pestañas.
- Opciones para el ancho de las pestañas:
  - Fijo: Valor indicado en el campo anterior.
  - Proporcional: Los lados de las piezas se dividen en el mismo numero de pestañas y hendiduras siendo el tamaño de las pestañas mayor o igual al ajuste al indicado en el campo anterior.
- Grosor del material.
- Anchura del corte.
- Capas y estilo nos permite escoger entre varias opciones para determinar la distribución de las piezas en el proyecto.
- Tipo de caja nos permite escoger cuantos lados deseamos quitar para que la caja no sea totalmente cerrada. Las opciones son:
  - Totalmente cerradas (6 caras).
  - Un lado abierto. Se suprime un panel de Largo x Ancho.
  - Dos lados abiertos. Se suprimen dos paneles adyacentes.
  - Tres lados abiertos. Se suprime una capa de cada panel.
  - Extremos opuestos abiertos.
  - Dos paneles solo, juntos por una de las aristas.
- Divisores en longitud permite crear piezas adicionales para montarlas dentro de la caja a lo largo del eje de longitud.

- Divisores en anchura permite crear piezas igual que el anterior pero estas se montarán en el eje de anchura.
- Espacio entre las partes indica el espacio entre las piezas en el dibujo.

Requisitos funcionales:

- RF1. Se creará una opción que permita que las piezas tengan todas sus líneas en un único *path*.
- RF2. Se eliminarán las coordenadas duplicadas de las esquinas en la nueva versión.

## B. Diseño

Se creará un nuevo checkbox llamado “*path*” de tipo bool de forma que si se encuentra activo se crearán las líneas de las piezas del cubo, cada una, en un único *path*.

Se guardará en una variable de texto la concatenación de los 4 conjuntos de coordenadas de cada uno de los lados de las piezas quitando el primer desplazamiento (para que se siga dibujando desde donde se quedó) y la última coordenada de tan solo el último tramo (para que no se dupliquen datos).

En la versión original se crea una nueva capa cada vez que se ejecuta la extensión pero no incluye nada. En la versión original esta capa se va a dejar de crear.

## C. Implementación

Para la implementación de esta función se ha tenido que añadir una nueva opción en la interfaz XML:

```
<param name="path" type="boolean" _gui-text="Just one path per shape">0</param>
```

Y en cada una de las llamadas a la función que genera las coordenadas se ha consultado si la variable *path* vale True para concatenar las coordenadas de los cuatro lados:

```
if path:
    sa=side((x,y),(d,a),(-b,a),keydivwalls*atabs*(-thickness if a else thickness),dx,(1,0),a,1,divx*yholes,yspacing,thickness)
    sb=side((x+dx,y),(-b,a),(-b,-c),keydivfloor*btabs*(thickness if b else -thickness),dy,(0,1),b,1,0,0,thickness)
    sc=side((x+dx,y+dy),(-b,-c),(d,-c),keydivwalls*ctabs*(thickness if c else -thickness),dx,(-1,0),c,1,0,0,thickness)
    sd=side((x,y+dy),(d,-c),(d,a),keydivfloor*dtabs*(-thickness if d else thickness),dy,(0,-1),d,1,0,0,thickness)
    total = sa + sb[sb.find('L'):]+ sc[sc.find('L'):] + sd[sd.find('L'):sd.rfind('L')]+ ' Z'
    drawS(total)
```

Una vez que se tienen todas las coordenadas en una única variable se llama al método `drawS` encargado de dibujar las líneas que se indiquen en la variable pasada como parámetro.

## D. Pruebas

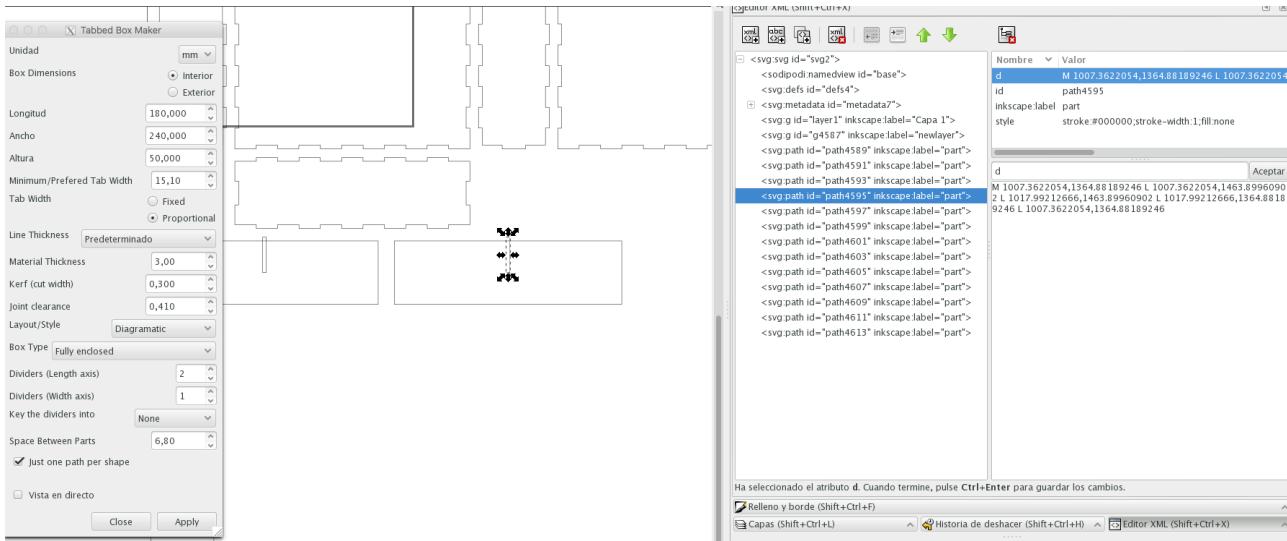
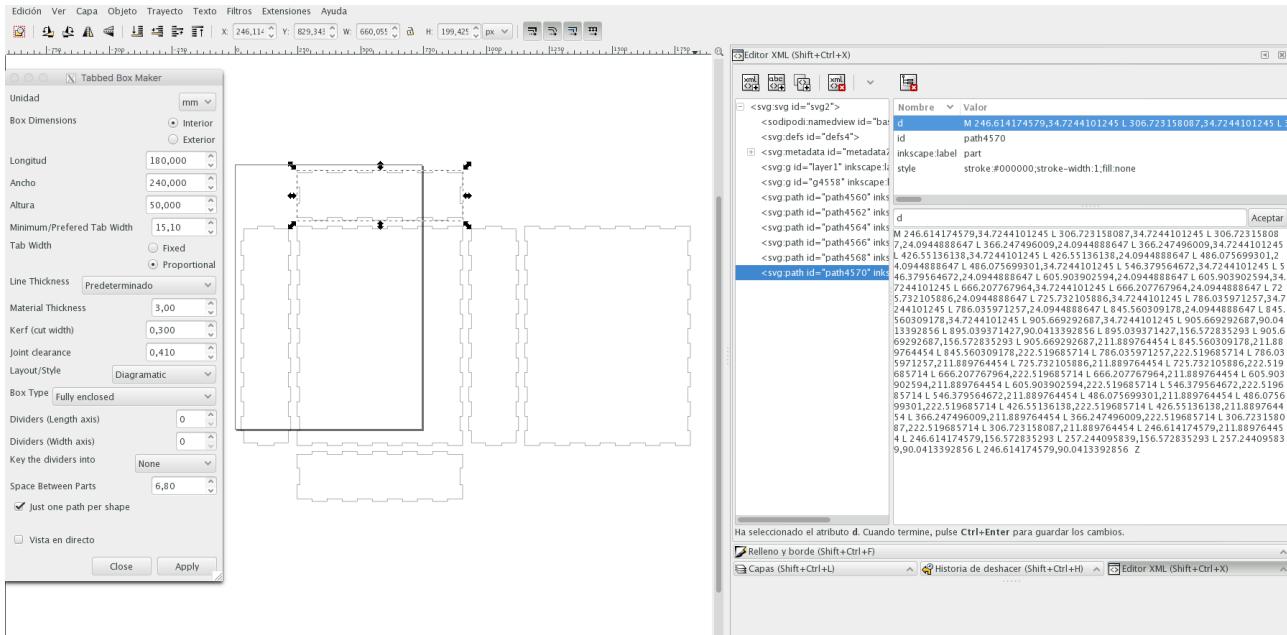
**Nombre:** Prueba 3-1.

**Descripción:** Comprobar que todos los bordes de una misma pieza se almacenan en un único `path`.

**Resultado:** SATISFACTORIO PARCIALMENTE.

**Descripción:** Se puede comprobar que todas las caras de la caja se almacenan en un único `path`, pero cuando se dibujan separadores internos con huecos para encajarlos se crean varios `path`, ya que, tal y como está implementado sería imposible reunirlos todos en uno.

**Estado final:**

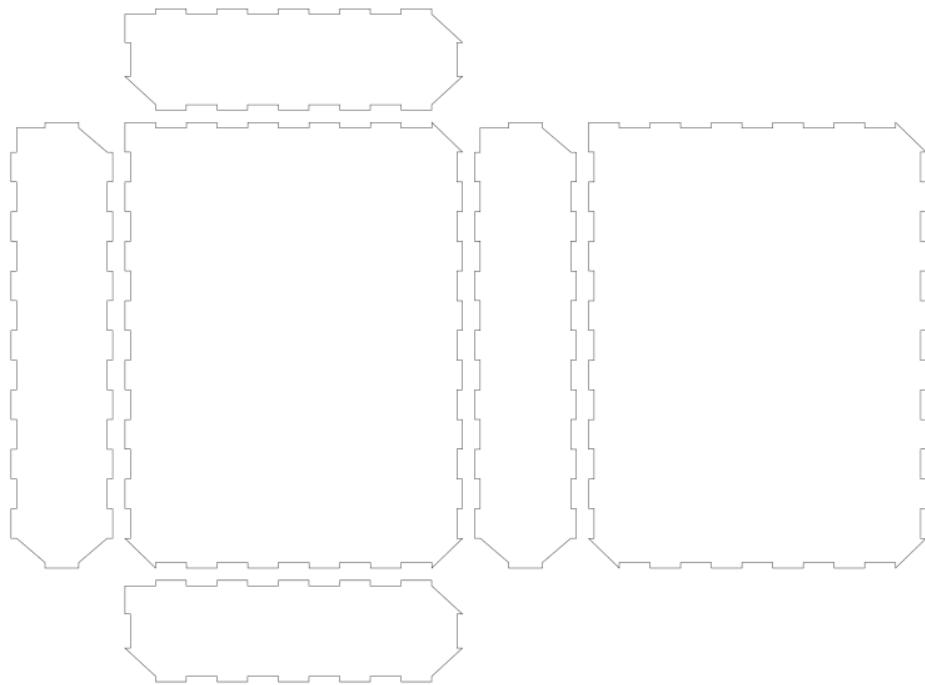


**Nombre:** Prueba 3-2.

**Descripción:** Comprobar que no se duplican esquinas.

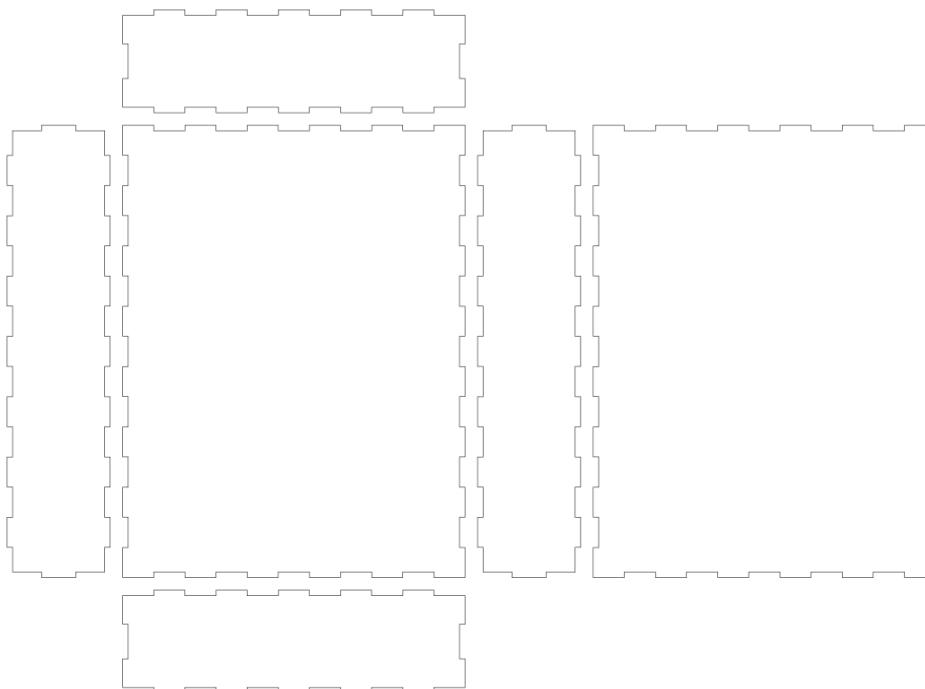
**Resultado:** FALLIDO.

**Descripción del problema:** Como se puede ver en la siguiente imagen, al eliminar la última coordenada de todas las paredes de las piezas vemos que en tres de las esquinas falta una coordenada y hace un entrante en vez de dibujar el pico correspondiente.



**Solución:** Como solución a este problema he optado por eliminar la última coordenada de tan solo la última línea o lateral y añadirle "z" para que cierre el polígono.

**Estado final:**



---

## 5.4. Sólidos platónicos

Se quiere crear una extensión para dibujar las caras de los 5 sólidos platónicos y cortarlas para encajarlas como un “puzzle3D”. Se introducirán como parámetros la longitud de las aristas, la anchura de pestaña, grosor del material, que será la distancia que sobresalgan las pestañas, una casilla para indicar si se desea calado o no calado y en caso afirmativo se facilitará la distancia desde la arista exterior a la interior del calado.

### A. Análisis

Requisitos funcionales:

RF1. Se podrá facilitar las medidas en cualquier unidad.

RF2. Se podrá seleccionar que sólido platónico queremos generar.

RF3. Como parámetros se introducirá la longitud de las aristas, la anchura de las pestañas y el grosor del material.

RF4. Se podrán modificar las pestañas que tiene cada lado oscilando sus valores entre 0 y 3.

RF5. De manera adicional se añadirá una casilla para que se pueda generar tan solo una pieza del sólido platónico seleccionado.

RF6. Se podrá escoger la distribución de las piezas, de manera que aparezcan todas en horizontal o se organicen de manera cuadrada/rectangular para aprovechar mejor el espacio.

RF7. Como el cliente nos indicó se creará un botón para indicar si las piezas serán caladas o no y se pondrá un campo que indique la distancia entre las aristas internas y externas.

RF8. Se pondrá una casilla para que todas piezas dibujadas se agrupen dentro de un grupo.

RF9. No se permitirá que el ancho de las pestañas sea mayor a la longitud de las aristas.

RF10. No se permitirá en caso de ser calado que la distancia entre las aristas internas y externas sea menor que el grosor. Esto es para que las pestañas no sobrepasen el corte del calado.

RF11. No se permitirá que la distancia que ocupan las pestañas por el numero de pestañas sea mayor que la longitud de las aristas.

RF12. Se podrá personalizar el estilo de las figuras seleccionando su color, en valores RGB y el grosor de los trazos.

Requisitos no funcionales:

RNF1. Deberá implementarse en python.

RNF2. La interfaz contará con diferentes parámetros.

RNF3. El diseño del código se hará totalmente modular para ayudar a una rápida comprensión.

RNF4. Se programará de manera que sea fácilmente adaptable a otros sólidos. Esto se consigue modularizando el código al máximo.

## B. Diseño

Para la implementación de esta extensión se va a buscar la máxima modularidad, por eso se han dividido los métodos en bloques:

Generadores de trozos:

- Generador de pestañas
- Generador de aristas

Generador de polígonos:

- Generador de triángulos
- Generador de cuadrados
- Generador de pentágonos

Generador de sólidos platónicos:

- Generador de sólidos formados por triángulos
- Generador de sólidos formados por cuadrados
- Generador de sólidos formados por pentágonos.

Utilizo la descripción anterior para señalar que cada uno de los métodos generadores señalados necesitan o dependen de los métodos generadores de niveles anteriores.

Por ejemplo, Generador de sólidos formados por triángulos depende de Generador de triángulos y este a su vez depende de Generador de aristas y generador de pestañas.

## C. Implementación

Los métodos que se han implementado buscando la máxima modularidad son:

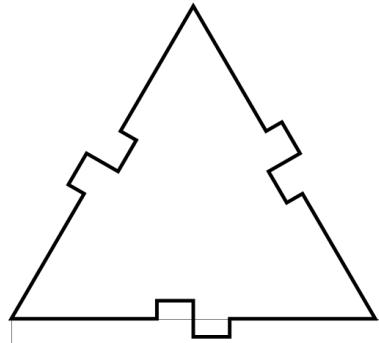
- *rgbToHex (r, g, b)*: Método encargado de convertir los tres parámetros (red, green, blue) a un color en hexadecimal que es lo que utiliza Inkscape.
- *creadorPestana (x1, y1, grado, grosor, anchoPestana)*: Método encargado de crear una pestaña (dobles ya que cada una cuenta con una parte entrante y una saliente) a partir de un punto dado en *x1* e *y1*. El parámetro *grado* nos indica el grado de inclinación que tiene la arista así que trabajo con ese grado y modificaciones de 90° y -90° para crear las pestañas de manera perpendicular a la arista. Grosor es la largura de las pestañas, es decir, lo que sobresalen o entran desde la arista y el *anchoPestana* es el grosor de la pestaña.



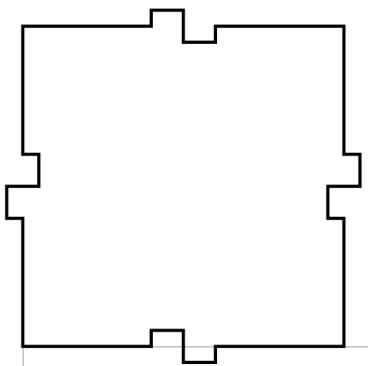
- *crearLado* ( $x1, y1, longitud, grado, grosor, anchoPestana, numPestana$ ): Este método es el encargado de generar las coordenadas de cada arista de los polígonos.  $x1$  e  $y1$  son los puntos desde donde parte la línea, *longitud* es el tamaño de la línea, *grado* será la inclinación de la línea, *grosor* nos indica la altura de las pestañas, *anchoPestana* se necesitará para llamar al método anterior para generar las pestañas y *numPestana* nos indica el número de veces que tendremos que llamar al método creador de pestañas.



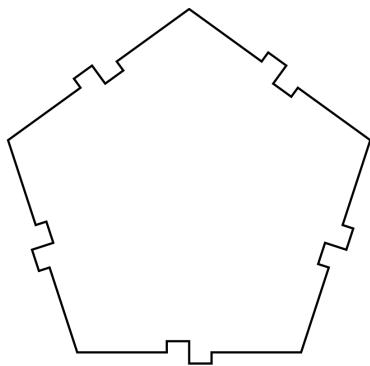
- *crearTriangulo* ( $x1, y1, longitud, grado, grosor, anchoPestana, numPestana, isCalado, distanciaCalado$ ): Método encargado de generar las coordenadas de un triángulo a partir del punto  $x1$  e  $y1$ , con una *longitud* de arista, un *grado* de inclinación (habrá triángulos en posición creciente y decreciente, invertidos), *grosor* del material para indicar la altura de las pestañas, *anchoPestana* que nos da el grosor de las pestañas, *numPestana* que da el numero de pestañas por arista, *isCalado* es una variable booleana que nos indica las caras serán caladas y en caso de ser así se utiliza *distanciaCalado* para indicar la distancia entre las aristas interiores y exteriores.



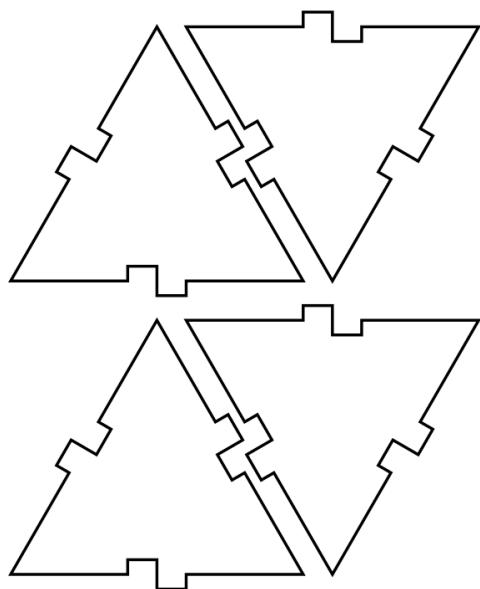
- *crearCuadrado* ( $x1, y1, longitud, grosor, anchoPestana, numPestana, isCalado, distanciaCaldado$ ): Este método es el encargado de generar las coordenadas de los cuadrados. Los parámetros son todos iguales al método de creación de triángulos a excepción de *grado* que este no tiene ya que todos los cuadrados se dibujarán en el mismo sentido o la misma orientación.



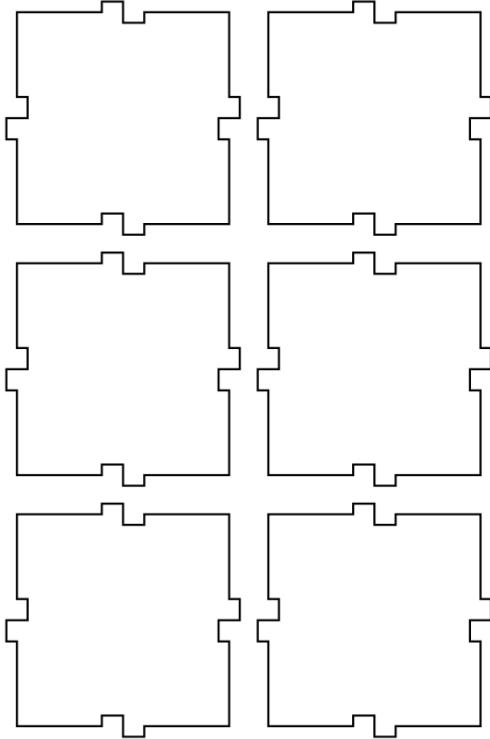
- *creadorPentagono* (*x1, y1, longitud, grosor, anchoPestaña, numPestana, isCalado, distanciaCalado*): Método encargado de generar las coordenadas de los pentágonos, sus atributos son iguales exactamente a los del creador de cuadrados ya que los pentágonos también se dibujarán todas con la misma orientación.



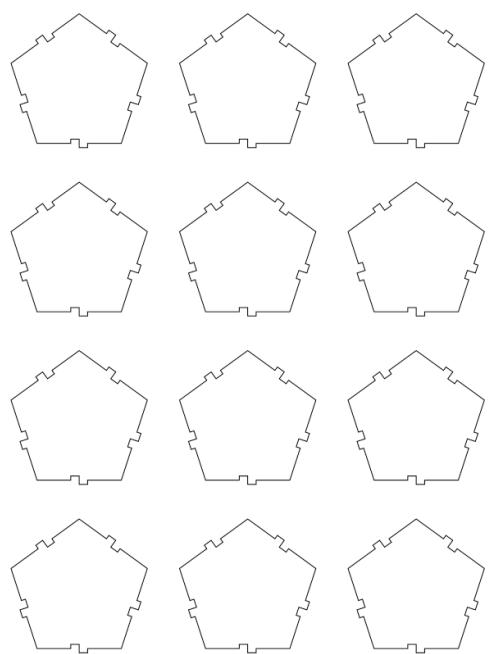
- *creadorSolidoTriangulo* (*longitud, grosor, anchoPestaña, numPestana, isCalado, distanciaCalado, distribucion, numero*): Este método se encarga de dibujar el conjunto de todos los triángulos que generan un sólido platónico (pueden ser tetraedro, octaedro o icosaedro). La mayoría de sus argumentos son iguales a los de los métodos anteriores menos *distribución* que indica la disposición de las piezas a la hora de dibujarlas y *numero* que indica el número de triángulos que se generarán (4, 8 o 20).



- *creadorSolidoCuadrado (longitud, grosor, anchoPestaña, numPestana, isCalado, distanciaCalado, distribucion, numero)*: Este método es el encargado de generar el hexaedro, compuesto por 6 caras cuadradas. Sus atributos son similares a los del creador de sólidos formados por triángulos pero este no cuenta con número ya que siempre va a dibujar 6 piezas y si cuenta con *separacion* que es la distancia que se dejará en blanco entre las piezas.



- *creadorSolidoPentagono (longitud, grosor, anchoPestaña, numPestana, isCalado, distanciaCalado, distribucion)*: Este método es el encargado de generar los dodecaedros. Todos sus atributos aparecen en los métodos anteriores y tienen la misma función.



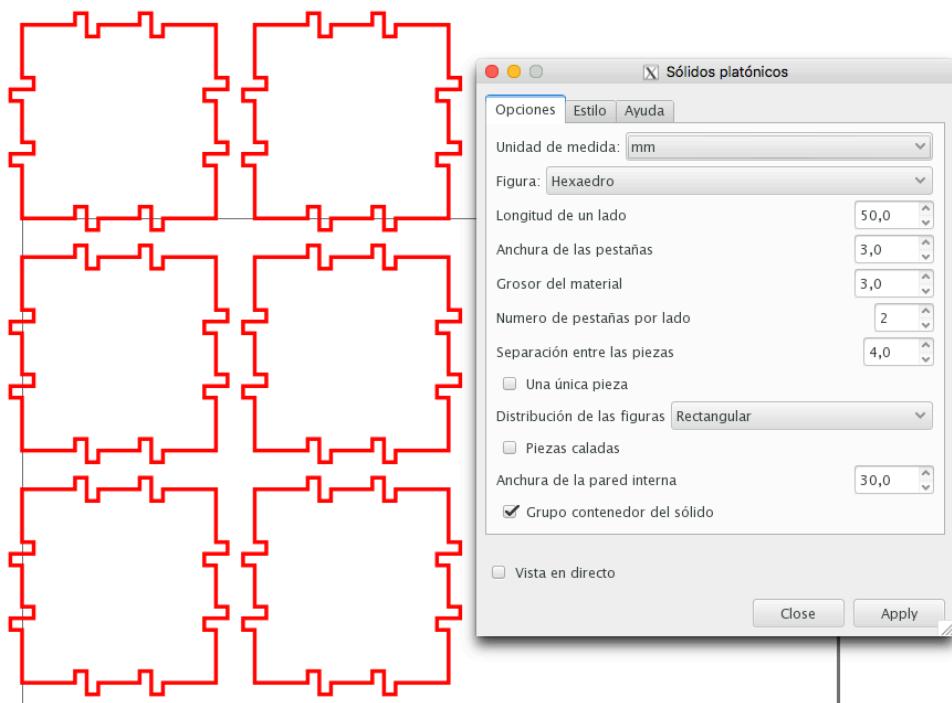
## D. Pruebas

**Nombre:** Prueba 4-1.

**Descripción:** Comprobar que se crean los sólidos platónicos (los 5 tipos) con una buena distribución rectangular.

**Resultado:** SATISFACTORIO.

**Estado final:**

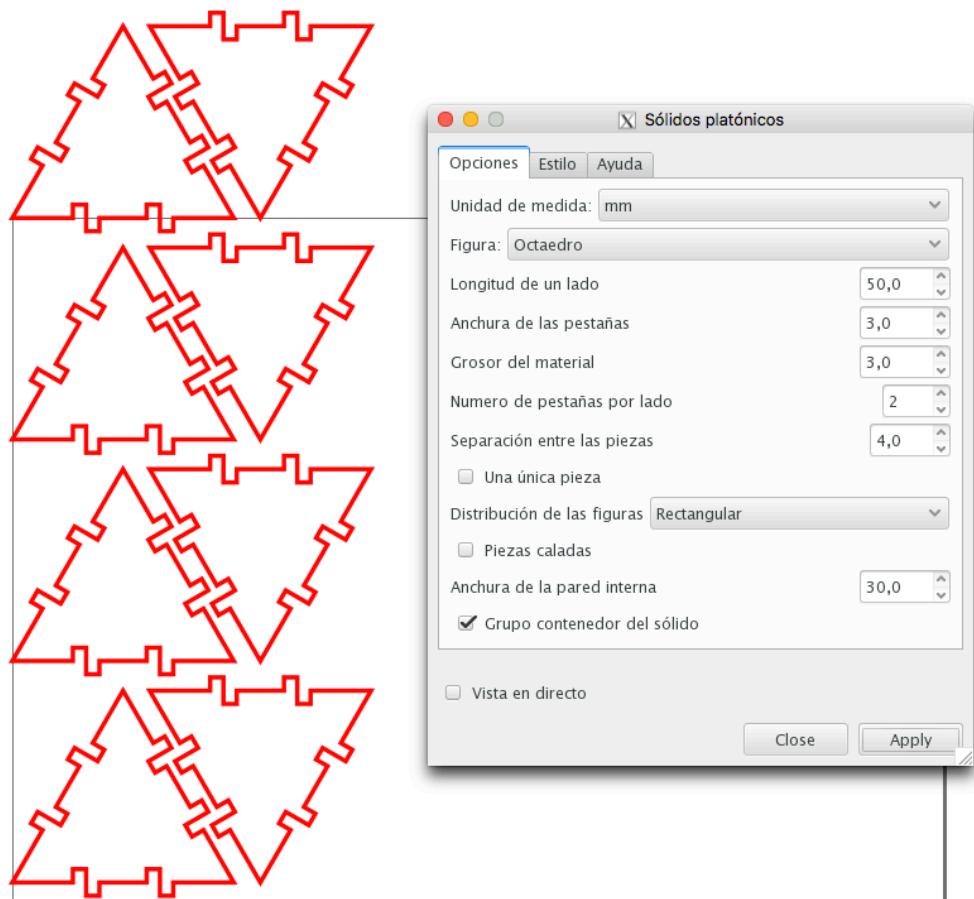


**Nombre:** Prueba 4-2.

**Descripción:** Comprobar que al dibujar un sólido platónico formado por triángulos estos se coloquen hacia arriba y hacia abajo de manera secuencial para aprovechar mejor el espacio.

**Resultado:** SATISFACTORIO

**Estado final:**

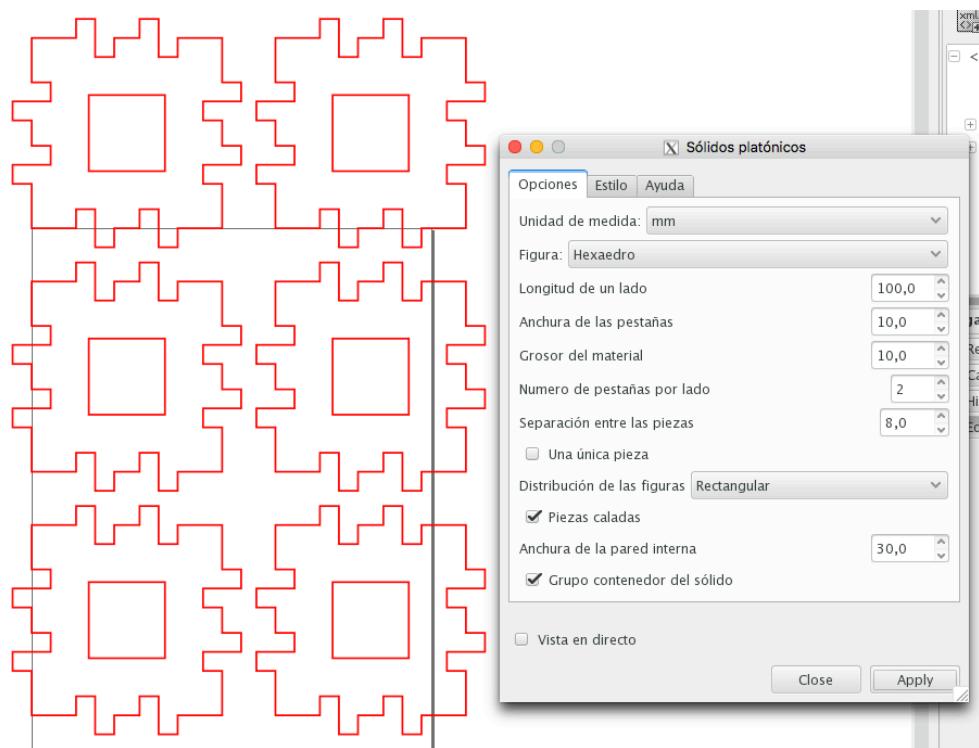


**Nombre:** Prueba 4-3.

**Descripción:** Comprobar que las piezas aparecen caladas si se selecciona la opción y que funciona correctamente la variable de distancia de pared interna.

**Resultado:** SATISFACTORIO

**Estado final:**

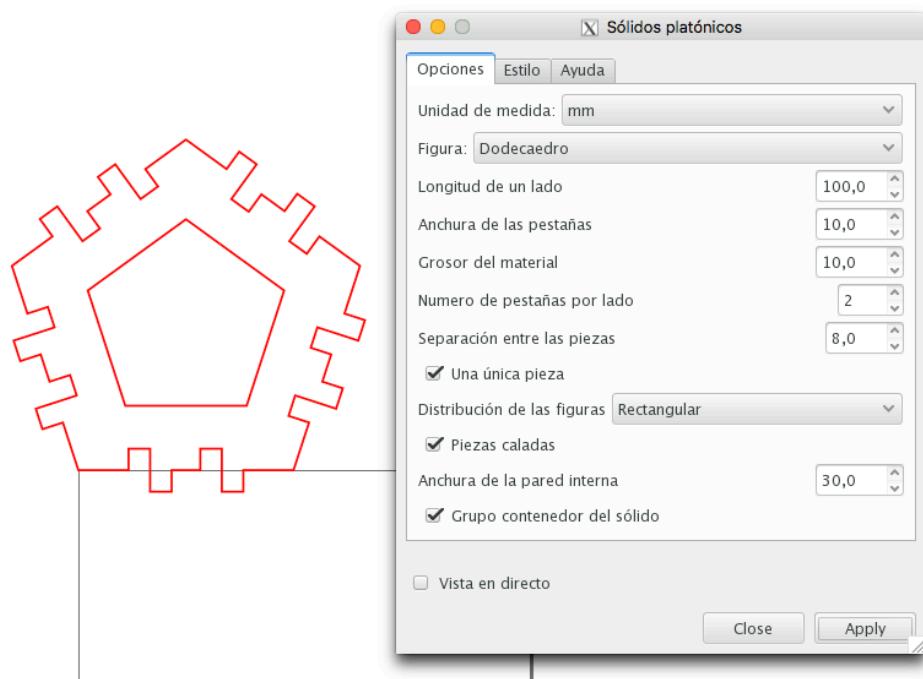


**Nombre:** Prueba 4-4.

**Descripción:** Comprobar que al marcar la casilla de una única pieza solo aparezca un pentágono, por ejemplo.

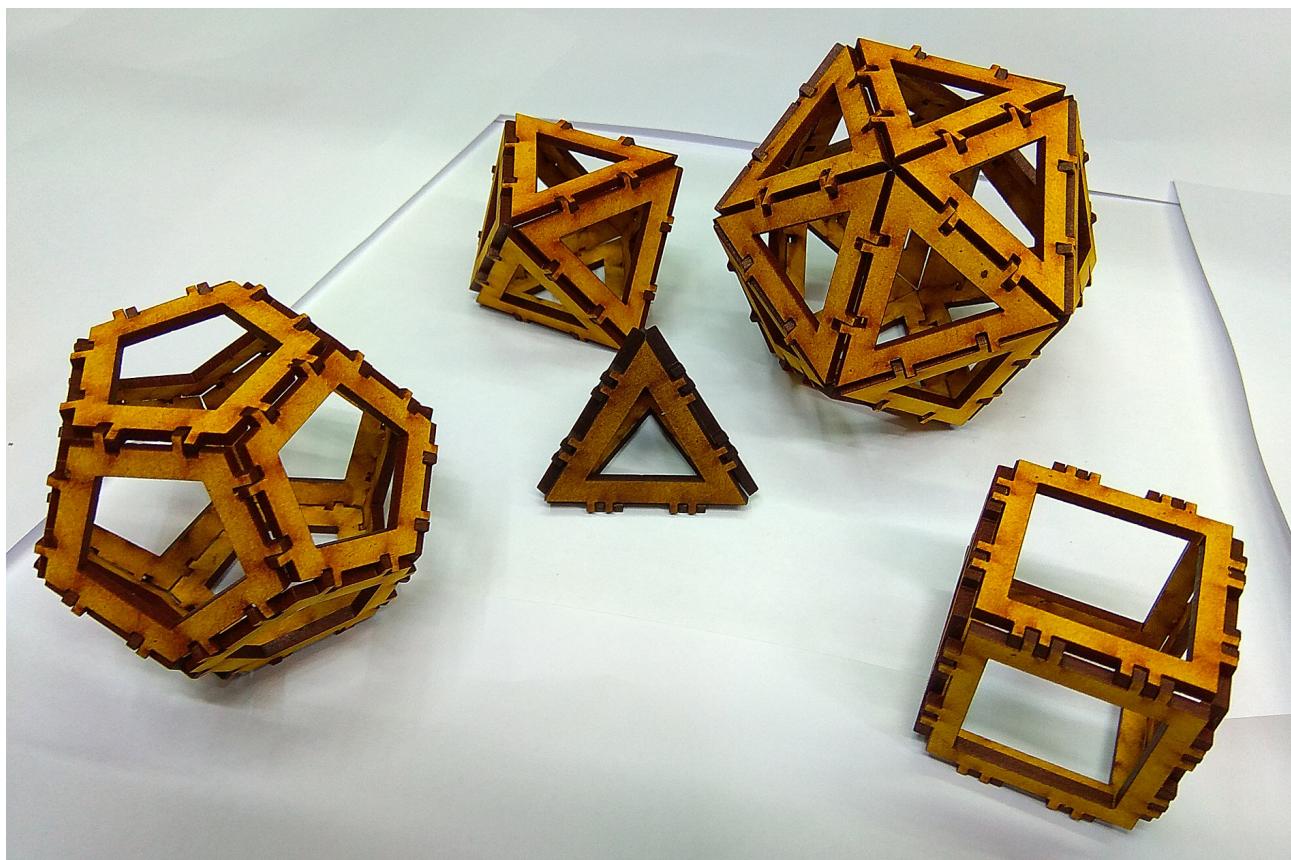
**Resultado:** SATISFACTORIO.

**Estado final:**



## E. Observaciones

Una vez que se terminó el diseño e implementación de esta función se realizaron algunas pruebas obteniendo como resultado los sólidos siguientes:



## 5.5. Ajuste sobre letras de un solo trazo

Por normal general los programas asumen las letras como polígonos cerrados rellenos de algún color pero para cortadoras láser, que es en lo que nos centramos, se ha tenido que diseñar un nuevo tipo de fuente.

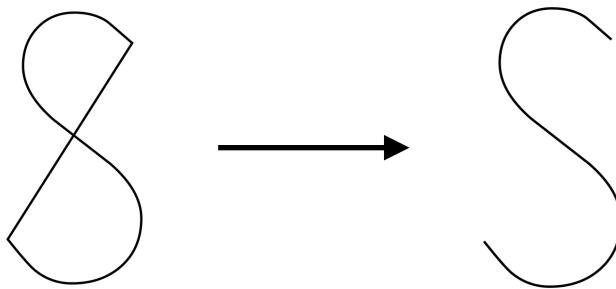


Esto se debe a que, por ejemplo, si queremos cortar la letra “A”, escrita en una fuente estándar, al terminar el trazo externo la pieza se movería y sería imposible recortar el interior con exactitud.

Estudiando este problema se creó un nuevo tipo de fuentes llamados *single line font*<sup>[12]</sup> en donde las letras están formadas por un único trazo, como si se dibujasen sin levantar el lápiz del papel.

El problema que hay que solucionar lo plantean los programas de dibujo, ya que asumen que los objetos de tipo texto son polígonos cerrados y cierran todas las letras que no empiecen y terminen el trazo en el mismo punto.

Por ejemplo la S empieza en un punto superior y va formando la figura hasta terminar en la parte de abajo. El programa, después de esto, cierra el polígono convirtiendo la letra en algo incomprensible e imposible de dibujar con la cortadora, por lo tanto habría que eliminar ese último trazo:



Para solucionar este problema existe una extensión llamada *Hershey text*<sup>[13]</sup> donde se puede escribir el texto que se desea dibujar con trazos únicos. El problema de esta extensión es que una vez que se dibuja el texto en cuestión no se puede modificar ya que se dibuja directamente en trazos, no en elementos de texto.

Otra posible solución a este problema ya había sido investigada e implementada, se encuentra en la red con nombre: *reopenSingleLineFont*<sup>[14]</sup>. Esta ha sido la solución escogida y sobre la que ha trabajado para ampliar su funcionalidad y adaptarla a las necesidades que el cliente nos expuso.

### A. Análisis

Requisitos funcionales:

RF1: Se tendrá que eliminar el último trazo en las letras para que sean comprensibles.

RF2: En caso de que no se seleccione ningún elemento antes de lanzar la extensión se devolverá un error informando de lo ocurrido.

Requisitos no funcionales:

RNF1: Se deberá implementar en python.

## B. Diseño

El diseño de esta extensión consiste en recorrer los puntos o coordenadas de cada una de las letras e ir estudiando si las líneas que conecta son necesarias o, si por lo contrario son una de las líneas que se deben borrar.

## C. Implementación

La extensión que se ha escogido como solución tenía dos versiones. Una para versiones muy antiguas y otras para actuales por lo que se ha tenido que depurar y "limpiar" el código y además de las funciones de las funciones con las que ya contaba se han añadido nuevas utilidades como la de aplicar los cambios a cada una de las letras aun encontrándose agrupadas.

## D. Pruebas

**Nombre:** Prueba 5-1.

**Descripción:** Se seleccionará un conjunto de letras (*paths*) y se ejecutará la extensión para ver si el resultado es correcto.

**Resultado:** SATISFACTORIO.

**Estado inicial:**

abade~~f~~ghijklmn  
ñopqrstuvwxyz

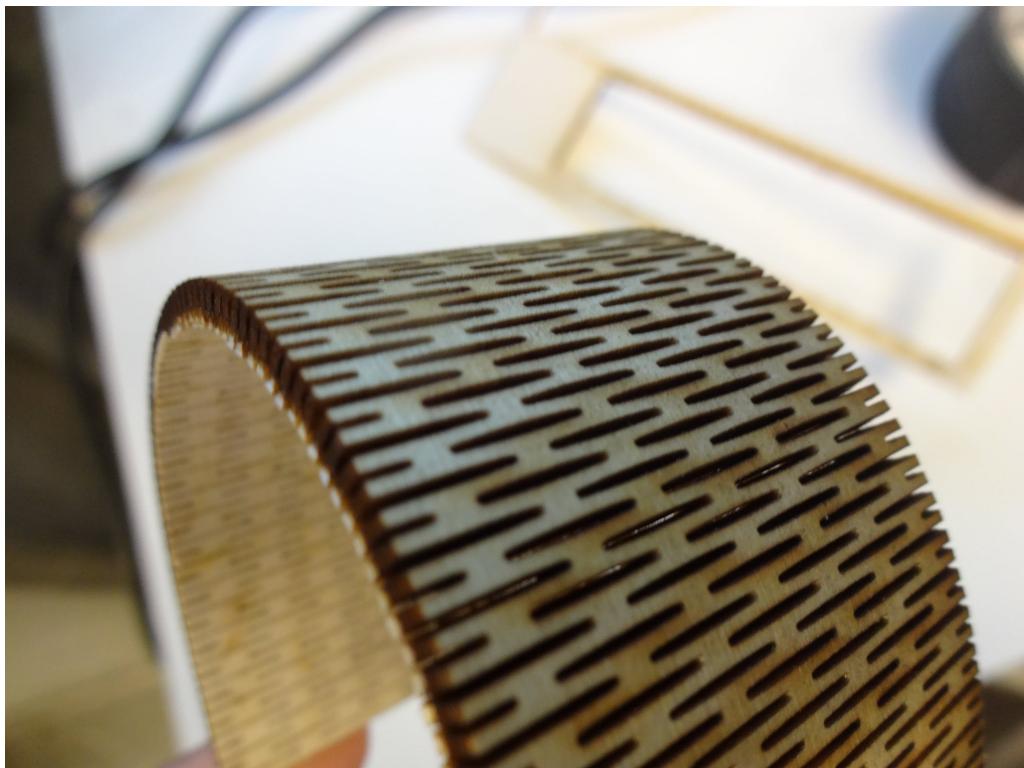
**Estado final:**

abcdefghijklmn  
ñopqrstuvwxyz

## 5.6. Patrón para doblar madera

El cliente solicita la creación de una extensión con la que generar un patrón formado por un conjunto de líneas para doblar planchas de madera

Esta técnica se denomina *láser bent wood*[15] y consiste en realizar una serie de cortes paralelos distribuidos de manera que aportan flexibilidad permitiendo doblar las planchas de madera. También se realizan patrones dibujando otras formas diferentes, pero en este caso se generará un patrón de líneas verticales como en la siguiente foto.



Para la generación del patrón el cliente indica que se deberá dar los siguientes parámetros o variables:

- Anchura del patrón
- Altura del patrón
- Separación horizontal entre las líneas
- Separación vertical entre las líneas
- Largura máxima de las líneas
- Opción de agrupar todas las líneas
- Opción de seleccionar si se quieren todas las líneas en un único *path* o en varios
- Distancia que sobrepasarán las líneas por el borde (para evitar problemas de no llegar en algunas cortadoras láser)
- Orden del dibujado de las líneas, se podrá escoger entre siempre de arriba a abajo o zigzag que agilizará el cortado.

## A. Análisis

Requisitos funcionales:

RF1. Las líneas tendrán que ser paralelas y siempre en horizontal pero no empezarán a la misma altura, deberán ir alterando el punto de comienzo.

RF2. Se tendrá que calcular el numero de líneas por columna para que en ningún momento se dibuje una línea superior a la largura máxima.

RF3. Se podrá configurar cualquier combinación de *path* (único o múltiple) y de orden de pintado (siempre de arriba a abajo o en zigzag).

RF4. Se dará la opción de dibujar una caja contenedora con la anchura y altura facilitada como parámetros con la que se podrá visualizar fácilmente los restos sobrantes que se ponen de excedente (no se agrupará con las líneas del patrón).

RF5. Nunca se podrá dibujar más allá de los laterales establecidos por el tamaño pero si se podrá pintar por fuera de los limites superior e inferior en caso de que se configure sobrante.

Requisitos no funcionales:

RNF1. Se deberá implementar en python.

## B. Diseño

Un primer diseño pensado para esta extensión consiste en calcular a partir de la altura del patrón, la longitud máxima de las líneas y la separación vertical entre ellas el número total de líneas que va a tener cada columna.

Una vez que se consigue digo valor se harán dos bucles (anidados) de forma que uno vaya creado o gestionando las columnas y el otro vaya pintando las líneas en cada una de ellas.

Para dibujar las líneas a partir de alturas diferentes he pensado en dibujar las pares a partir del eje x, y y las impares a partir de la x,  
 $y + \text{mitadLongitudLínea} + \text{separaciónVertical}/2$ .

## C. Implementación

En la implementación de esta extensión se han codificado dos métodos intentando extraer partes del código que se iban a repetir bastante en cualquier ejecución.

Estos métodos son:

- *draw\_SVG\_line (x1, y1, largura, altura, especial, sobrante, inverso)*: Este método es el encargado de generar las coordenadas de las líneas, siempre en vertical, a partir de un punto dado por *x1* e *y1*. *largo* indicará la longitud de la línea mientras que *altura* será el valor pasado por la interfaz como altura del patrón. Especial es un parámetro que nos indica (solo en el caso de las líneas impares) el punto de comienzo de la línea (la primera de la columna). Por ultimo sobrante es la distancia que el usuario quiere que sobresalga de la caja contenedora máxima (para asegurar que la cortadora corta más allá del borde de la plancha) e inverso nos indica cuando estamos dibujando líneas de abajo a arriba para el caso de dibujo en zigzag.

- `boxPoints(anchura, altura)`: Este método es el encargado de generar las coordenadas de la caja contenedora a partir de la coordenada 0,0 y con la *altura* y *anchura* pasadas como parámetro.

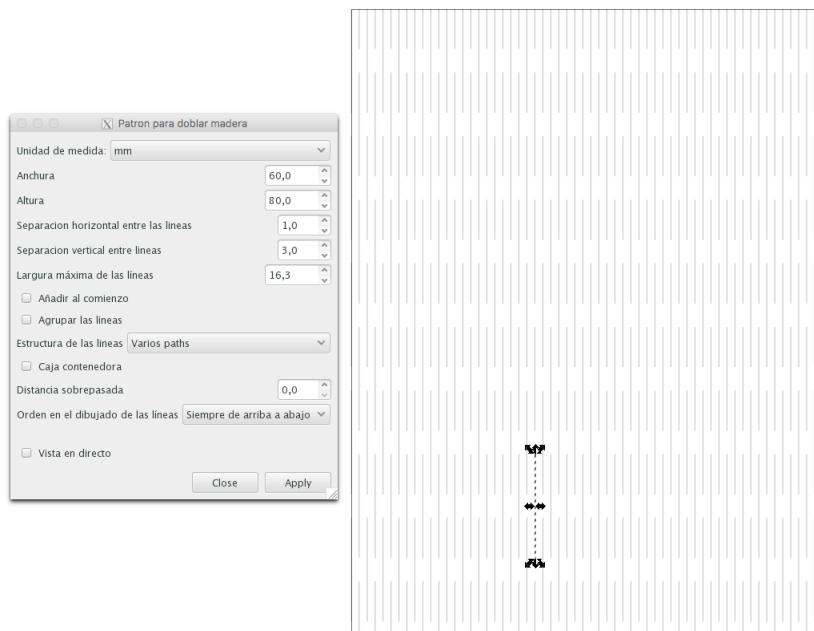
#### D. Pruebas

**Nombre:** Prueba 6-1.

**Descripción:** Comprobar que se crea un patrón correctamente. (Dibujando siempre de arriba hacia abajo y creando un *path* para cada una de las líneas).

**Resultado:** SATISFACTORIO.

**Estado final:**

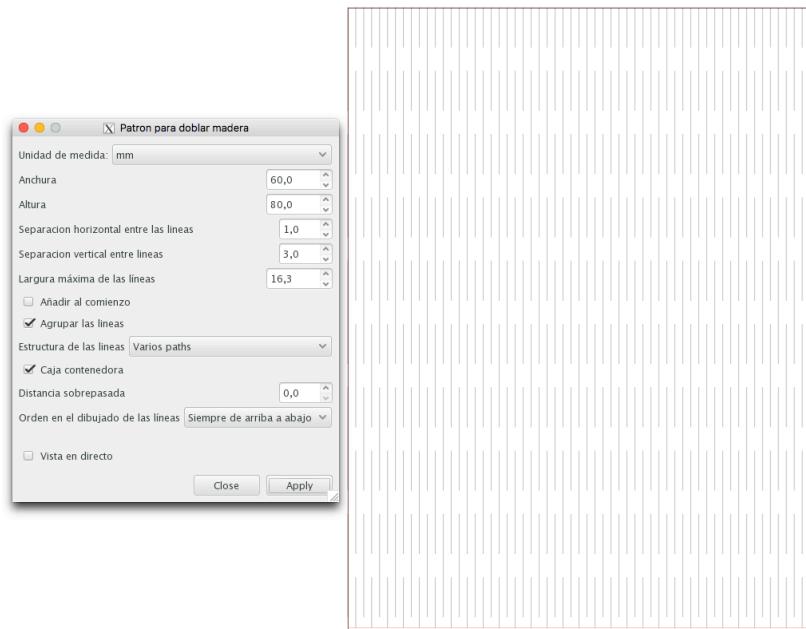


**Nombre:** Prueba 6-2

**Descripción:** Sobre la prueba anterior probar a dibujar la caja contenedora y agrupar todas las líneas.

**Resultado:** SATISFACTORIO.

**Estado final:**

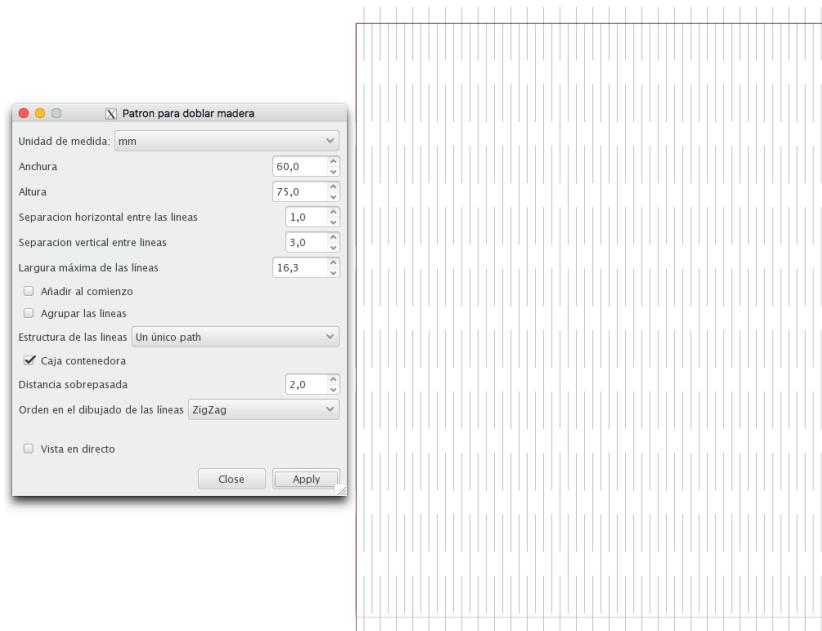


**Nombre:** Prueba 6-3

**Descripción:** Dibujar un patrón donde todas las líneas compartan un único path, se dibujen en orden zigzag y tenga un sobrante de 2 mm

**Resultado:** SATISFACTORIO.

**Estado final:**



---

## 5.7. Exportar capas a EPS

El cliente nos solicita que creemos una extensión capaz de exportar cada una de las capas de un proyecto en archivos con extensión EPS diferentes.

Esto no es un problema aislado, hay muchas cortadoras láser que necesitan separar en documentos EPS diferentes aquellas piezas que tengas características de corte diferente.

El cliente además cita que suele tener problemas con la cortadora si los archivos exportados tienen algunas características de diseño, por lo que nos especifica que hagamos ciertos cambios antes de generar los archivos EPS:

- La opacidad de los bordes deberá estar al 100%.
- Quitar el relleno de las figuras.
- Pasar los textos a *path*.

Todas estas opciones serán optativas por lo que habrá que añadir *checkboxs* en la interfaz para que el usuario pueda seleccionarlas o no.

### A. Análisis

Requisitos funcionales:

RF1. Todas las capas del documento se exportarán en formato EPS

RF2. Los ficheros generados tendrán el mismo nombre que la capa que exporta.

RF3. Los ficheros generados se crearán en el directorio que se haya indicado en el cuadro de la interfaz de la extensión. Siempre habrá que indicar rutas absolutas.

RF4. En caso de que se elimine el relleno a algún elemento de texto se mostrará un aviso por pantalla advirtiendo al usuario de que si no este no tiene borde se hará transparente.

Requisitos no funcionales:

RNF1. Se deberá implementar en python.

### B. Diseño

El procedimiento que se ha pensado para este ejercicio consiste en recorrer dos veces todos los elementos que tenemos a partir del nodo principal en caso de que se quiera poner la opacidad de los bordes al máximo o eliminar el relleno. En caso de que no se quiera hacer ninguna de esas dos funciones solo se recorrerán una vez.

La primera iteración por los elementos la he utilizado para eliminar el relleno o ajustar la opacidad del borde mientras que en la segunda vuelta nos encargamos de exportar las capas en formato eps (con más o menos argumentos dependiendo de si queremos pasar los elementos de texto a *path* o no).

### C. Implementación

Algunas de las líneas de código más interesantes que nos encontramos en esta extensión son aquellas en las que llamamos a la línea de comandos para poder realizar la exportación:

```
parameters = [
    "inkscape",
    INKSCAPE_WITHOUT_GUI,
    INKSCAPE_EXPORT_EPS, dest,
    INKSCAPE_EXPORT_ID, objeto.get('id'),
    INKSCAPE_EXPORT_AREA_DRAWING,
    INKSCAPE_EXPORT_FILTERS,
    INKSCAPE_EXPORT_TEXT,
    svg_file_path,
]
p =
subprocess.Popen(parameters, stdout=subprocess.PIPE, stdin=subprocess.PIPE, stderr=
=subprocess.PIPE)
    (stdoutdata, stderrdata) = p.communicate()
    p.stdout.close()
    p.stdin.close()
    p.stderr.close()
```

Podemos ver como se configuran todos los parámetros que se le pasarán a la línea de comandos y como se abre una comunicación con esta.

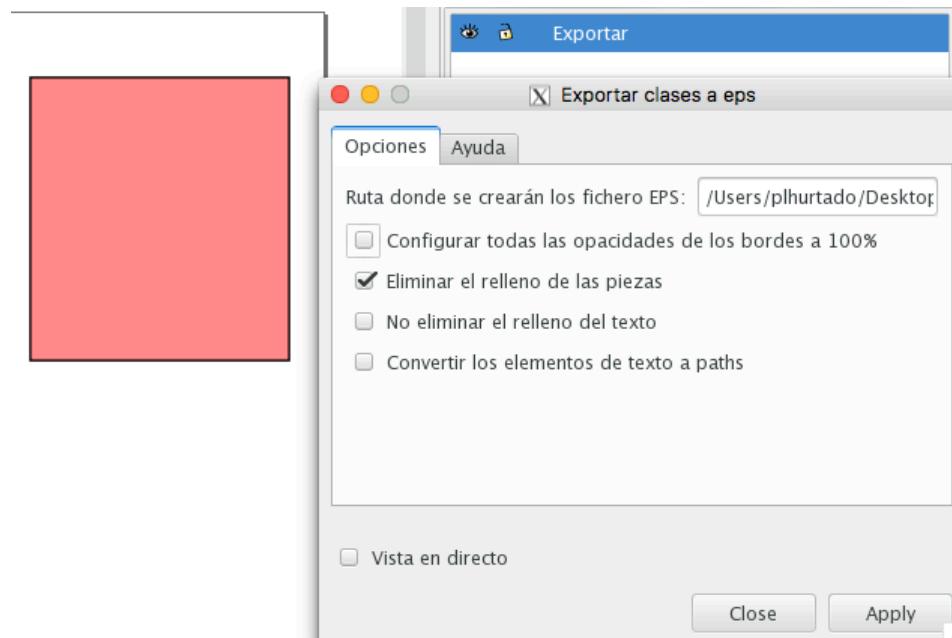
## D. Pruebas

**Nombre:** Prueba 7-1.

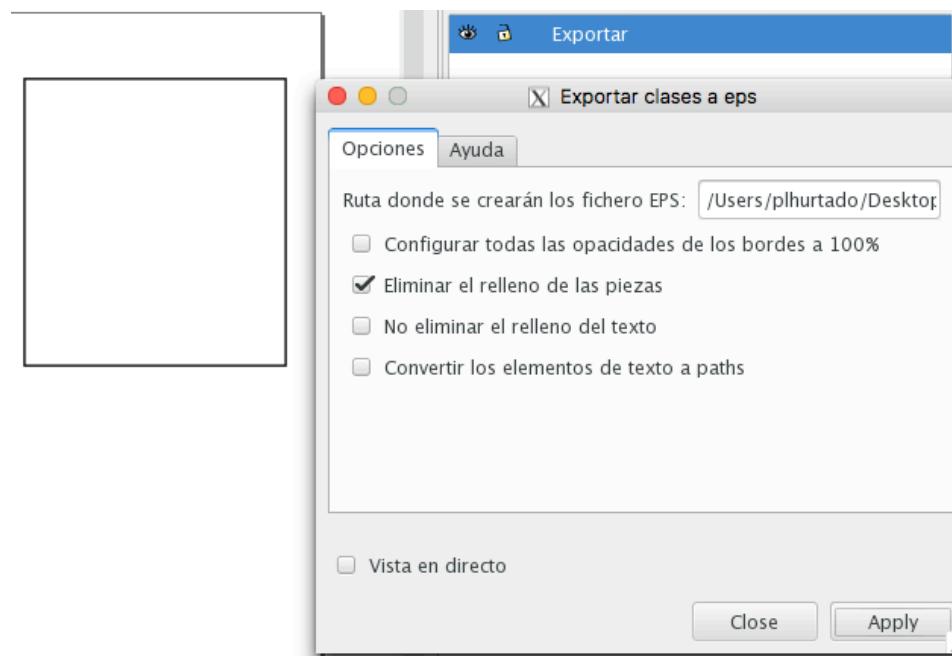
**Descripción:** Comprobar que se crea un archivo EPS en la ruta indicada eliminando el relleno de una figura.

**Resultado:** SATISFACTORIO.

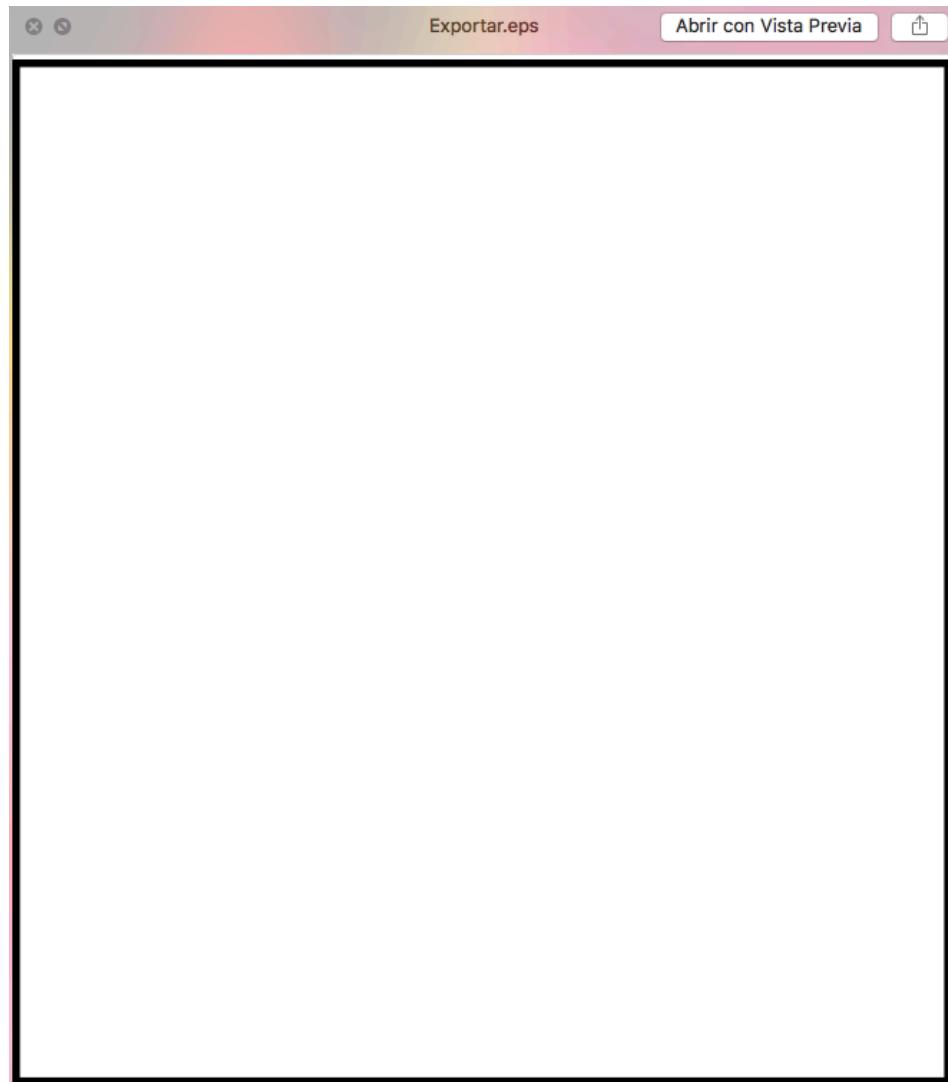
**Estado inicial:**



**Estado final:**



**Documento generado:**

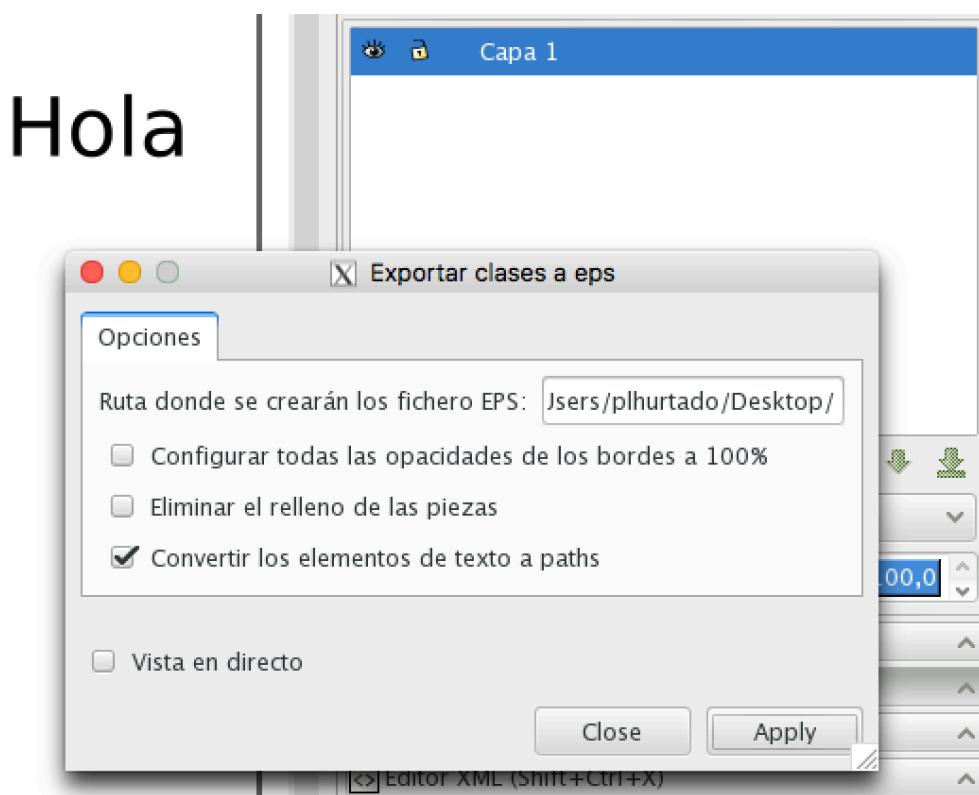


**Nombre:** Prueba 7-2.

**Descripción:** Corroborar que el archivo exportado se crea en la ruta señalada y que los objetos de texto se pasan a *path*.

**Resultado:** SATISFACTORIO.

**Estado inicial:**



**Estado final:**





## 6. CONCLUSIÓN Y TRABAJOS FUTUROS

Como se ha comentado en otros puntos de este documento, los objetivos principales del proyecto son crear una documentación útil, para gente que empiece a trabajar en este campo, y diseñar y crear o mejorar extensiones que puedan facilitar el trabajo a personas que utilizan este programa para el uso de cortadoras láser.

El apartado de creación de extensiones no se centra tan solo en la realización de nuevas extensiones desde cero, algunas de ellas requerían investigación y aprendizaje previo para mejorar extensiones ya creadas por otros desarrolladores.

Para resumir el trabajo realizado indicaré que objetivos se han completado totalmente:

1. Documentación para crear nuevas extensiones.  
Se han completado todos los puntos.
2. Capas por color.  
Se han completado todos los puntos.
3. Color por capa  
Se han completado todos los puntos.
4. TabbedBoxMaker  
Se han completado todos los puntos.
5. Sólidos platónicos  
Se han completado todas los puntos menos 4.9 donde se pedía escoger el color de los sólidos de una manera cómoda e intuitiva.
6. Ajuste sobre letras de un solo trazo  
Se han completado todos los puntos.
7. Patrón para doblar madera  
Se han completado todos los puntos.
8. Exportar capas a EPS  
Se han completado todos los puntos.

Como se puede ver, la gran mayoría de los puntos obligatorios que estaban definidos para este proyecto se han conseguido realizar. Uno de los puntos optativos de una extensión obligatoria (4.9) no he conseguido llegar a desarrollarlo por no encontrar información suficiente sobre como realizar lo que se pedía: En particular parece que no hay manera en Inkscape de que uno de las variables de la interfaz de parámetros sea de tipo color, hay que dar tres valores RGB para cada color.

Como trabajos futuros se han dejado los tres objetivos opcionales.

Los dos primeros son extensiones que requieren el uso de algoritmos de optimización (como por ejemplo, algoritmos genéticos) y el estudio y desarrollo de esto supera el tiempo disponible para desarrollar el proyecto, por eso decidí centrarme en los objetivos obligatorios e intentar implementarlos de la mejor manera posible. No obstante creo que, tanto la documentación generada como los archivos de código creados, son una base bastante estable para empezar a investigar a partir de ellos para conseguir estas metas.

La tercera actividad optativa consistía en internacionalizar el proyecto. Para esto hay que introducir en los archivos de traducción las frases o palabras que se iban a usar en inglés junto con su traducción al español. Me he focalizado mayormente en las actividades obligatorias y el resto de tiempo no ha sido suficiente para llegar a conseguir este objetivo.

En cuanto a aprendizaje y experiencia personal señalaré que toda la información relacionada con la creación y edición de extensiones para, en este caso, Inkscape es muy escasa. Esto dificulta enormemente la toma de contacto y primeras etapas de aprendizaje de los que investigamos en este campo.

Debido a esta dificultad para encontrar información útil y bien estructurada, uno de los objetivos obligatorios del proyecto era realizar una guía que puedan utilizar los programadores interesados en este sector, intentando facilitarles esas primeras etapas que a mi tanto me han costado.

En cuanto a la organización y estimación inicial de tiempos y costes hay cierto desfase, ya que ha quedado muy corto el tiempo previsto para la formación, investigación e implementación con respecto al invertido realmente.

Una vez que empecé a notar este retraso o ralentización tuve que plantearme modificar la fecha de entrega aplazándola a septiembre, aunque inicialmente la defensa iba a ser en julio.

La diferencia entre el tiempo estimado (82 horas) y el tiempo invertido realmente (140 horas) asciende a unas 58 horas que traducido a ganancias económicas supone una gran pérdida. Debido a que ésta es la segunda estimación temporal que realicé sobre un proyecto de mediano tamaño aún no cuento con la habilidad suficiente para hacer buenas estimaciones.

A nivel personal considero que este proyecto ha sido una de las tareas o trabajos que más me ha formado a lo largo de toda la carrera. Este hecho creo que se debe a la metodología de trabajo que suponen los “Trabajos fin de grado”, mucho más autónoma que el resto de actividades realizadas a lo largo de la carrera. El enfrentarse en solitario a un trabajo de investigación y desarrollo, por supuesto siempre guiado y orientado por el tutor, permite adquirir conocimientos mucho más asentados y estructurados, ya que son obtenidos a partir de los fallos y aciertos que uno mismo realiza.

Por último me gustaría mencionar el reto que este proyecto ha sido para mí.

Cuando el tutor me explicó el objetivo principal, lo escogí rápidamente entre las demás propuestas. Me daba la oportunidad de aprender un nuevo lenguaje de programación que no había usado en ninguna de las asignaturas cursadas y que ahora está siendo tan utilizado en el mundo de la programación, Python.

Ahora puedo afirmar que no me equivoqué eligiendo este proyecto entre los demás. El inicio fue complicado y tuve dudas sobre si haber escogido éste proyecto, basado en un lenguaje de programación desconocido para mí, había sido una buena idea. Hoy, con esta entrega, puedo decir que no me equivocaba y que el esfuerzo y horas dedicadas han merecido la pena, tanto por dejar una base para futuros compañeros que puedan seguir ampliándola como para mí, por tratarse de un reto personal que he podido superar gracias a lo aprendido durante estos 4 años y al apoyo de mi tutor.



## 7. ANEXO: PARTES DE LAS REUNIONES CON EL CLIENTE

Cómo anexos se adjuntarán los partes que he podido ir generando a partir de las reuniones o tutorías que he ido realizando con mi coordinador.

---

### 7.1. Parte 1

#### **Detalles**

- Número de reunión: 1
- Duración: 1 hora.

#### **Puntos a examinar**

1. Metodología de trabajo que se usaría durante el desarrollo del proyecto.
2. ¿Que lenguaje de programación se utilizará?.
3. ¿Como compartimos los documentos para que el profesor pueda tener acceso a ellos?.
4. Por donde empezar a desarrollar el proyecto.

#### **Discusiones y decisiones**

- En cuanto a la metodología que se usará para el desarrollo del proyecto el profesor ha propuesto alguna de las denominadas “ágiles” pero haciendo hincapié en que el TFG es un trabajo que se realiza durante un curso académico y que se tiene que compaginar con más asignaturas, por lo que sería bueno alguna forma de trabajo que tuviera suficientes iteraciones con el (reuniones) pero no demasiadas ya que podría ser un problema a la hora de cumplirlas.

Debido a esto he decidido proponer una metodología Scrum en la que los Sprints serán de 4 semanas o 1 mes siendo relativamente flexibles dichas fechas en caso de que sea necesario añadir algún tiempo extra o adelantar la reunión de fin de Sprint en caso de que se termine el trabajo indicado antes de tiempo.

- El lenguaje que se utilizará en este proyecto será principalmente Python por lo que he tenido que asignar un tiempo a aprendizaje ya que nunca había utilizado este lenguaje
- La manera que se utilizará para compartir toda la documentación y los archivos de código que se vayan creando será a través de Github (para los documentos de programación) y a través de Dropbox (para toda la documentación, actas de reunión, etc).
- El tutor propuso que se empezase pensando una organización primera o provisional para el desarrollo del proyecto, estudiando los diferentes aspectos en los que se tiene que profundizar un poco más (como son Python o el funcionamiento o estructura de los documentos se van a crear) y una vez que se tuvieran los puntos anteriores planteados empezar con la ingeniería de requisitos “provisional”.

#### **Adjuntos**

No se adjuntaron documentos en esta iteración.

---

## 7.2. Parte 2

### Detalles

- Número de reunión: 2
- Duración: 1 hora, 15 minutos.

### Puntos a examinar

1. El principal motivo de esta tutoría era solicitar ayuda con respecto a la gestión o manejo de las librerías de Inkscape ya que es muy difícil encontrar información bien detallada en internet.
2. También se pretendía llegar a una organización general de cuales son las extensiones que se van a realizar y cual será el orden de estas dependiendo del tiempo y su dificultad.
3. Hemos hablado también sobre la estructura de la documentación a grandes rasgos y como la debería desarrollar.

### Discusiones y decisiones

- Hemos llegado al acuerdo de realizar la primera extensión estando juntos para que me pueda ayudar un poco a ver como funciona la librería de Inkscape.
- Elaboraré una especie de documentación donde se explique la estructura general o formato de los documentos necesarios para las extensiones (XML y Python).

### Adjuntos

No se adjuntaron documentos en esta iteración.

---

## 7.3. Parte 3

### Detalles

- Número de reunión: 3
- Duración: 3 horas.

### Puntos a examinar

1. Aprender cual es la estructura interna en la que se almacenan los datos.
2. Trabajar de manera conjunta en la primera extensión para ver cuales son las principales funciones y métodos de las librerías.

### Discusiones y decisiones

- Se realizará una tutoría semanalmente de manera que se pueda ir viendo el progreso poco a poco y se puedan ir definiendo nuevas metas.
- Se realizará una documentación donde se explique cual es el formato de los documentos .inx y .py.
- Se realizará una documentación de las funciones y métodos de las librerías utilizadas para que les sea más fácil comenzar a personas que quieran crear extensiones para inkscape.

### Adjuntos

- Primera extensión de pruebas
- Enlaces de interés:
  1. <http://tavmjong.free.fr/INKSCAPE/MANUAL/html/CommandLine.html>
  2. <https://www.w3.org/TR/SVG/>
  3. <https://www.w3.org/TR/SVG/paths.html>
  4. <https://pypi.python.org/pypi/svgpathtools/1.2.5>

---

## 7.4. Parte 4

### **Detalles**

- Número de reunión: 4
- Duración: 2 horas, 30 minutos.

### **Puntos a examinar**

1. Comprobación y corrección de posibles fallos en la primer extensión realizada.
2. Planteamiento por parte del profesor de como debo realizar la documentación general y particular para cada una de las distintas extensiones.
3. Lista de extensiones que se van a realizar en una primera etapa.

### **Discusiones y decisiones**

- Debate sobre el tipo de análisis de requisitos que he de realizar.
- Como explicar el diseño de cada una de las extensiones. Se realizará a través de pseudocódigo.

### **Adjuntos**

No se adjuntaron documentos en esta iteración.

---

## 7.5. Parte 5

### **Detalles**

- Número de reunión: 5
- Duración: 1 hora, 30 minutos.

### **Puntos a examinar**

1. Comprobar el funcionamiento de los sólidos platónicos sin calado.
2. En capas por color pregunto como seleccionar los grupos de objetos (que no sean capas).
3. Resto de extensiones que se gestionarán en el proyecto.

### **Discusiones y decisiones**

- En capas por color, en caso de que se eliminen las capas originales se creará una nueva de donde cuelgue todo.
- En capa por color no se desharán los grupos de objetos en el que todos tengan el mismo color de borde.
- En sólidos platónicos se dará la opción de escoger la distribución de las piezas.
- En color por capa en caso de que el nombre de la capa no sea un color definido se le concatenará el nombre o valor del color en cuestión.

### **Adjuntos**

No se adjuntaron documentos en esta iteración.

---

## 7.6. Parte 6

### **Detalles**

- Número de reunión: 6
- Duración: 1 hora, 30 minutos.

### **Puntos a examinar**

1. ¿En qué consiste la extensión que permite doblar láminas de madera?
2. ¿Qué son las letras de un solo trazo?
3. ¿Cómo puedo crear una extensión que elimine los trazos inútiles de las letras de un único trazo o fuente máquina?

### **Discusiones y decisiones**

- La extensión para generar patrones que faciliten la curvatura en láminas de madera se creará desde cero y se le darán un conjunto de parámetros de manera que todo sea configurable.
- También tendrá una opción que generará las líneas en zig zag de manera que se pueda optimizar el tiempo que se pierde si siempre dibujas/cortas en el mismo sentido.
- Como base para la extensión sobre letras de un solo trazo se utilizará una facilitada por el profesor y se mejorará.

### **Adjuntos**

No se adjuntaron documentos en esta iteración.

---

## 7.7. Parte 7

### **Detalles**

- Número de reunión: 7
- Duración: 1 hora.

### **Puntos a examinar**

1. Corrección y detalles sobre la documentación

### **Discusiones y decisiones**

- Se añadirán las tutorías como anexos a la documentación.
- En todos los ficheros de código se tendrá que indicar el tipo de licencia que se ha utilizado.

### **Adjuntos**

No se adjuntaron documentos en esta iteración.



## 8. REFERENCIAS

- [1] **Wikipedia:** Laser cutting **URL:** [https://en.wikipedia.org/wiki/Laser\\_cutting](https://en.wikipedia.org/wiki/Laser_cutting) (**Última fecha de acceso:** 10/04/2017)
- [2] **Cosmocax:** Control numérico o CNC **URL:** <https://cadcamcae.wordpress.com/2007/06/14/el-control-numerico-por-computadora-el-cnc/> (**Última fecha de acceso:** 30/07/2017)
- [3] **Wikipedia:** G-code **URL:** <https://en.wikipedia.org/wiki/G-code> (**Última fecha de acceso:** 20/05/2017)
- [4] **Adobe:** Rasterizado y vectorizado **URL:** <https://helpx.adobe.com/es/photoshop-elements/key-concepts/raster-vector.html> (**Última fecha de acceso:** 20/05/2017)
- [5] **Velneo:** Formato SVG **URL:** <https://velneo.es/que-es-el-formato-svg/> (**Última fecha de acceso:** 02/06/2017)
- [6] **Inkscape:** Información sobre Inkscape **URL:** <https://inkscape.org/es/acerca-de/resumen/> (**Última fecha de acceso:** 02/06/2017)
- [7] **Wikipedia:** Licencia de software libre **URL:** [https://es.wikipedia.org/wiki/GNU\\_General\\_Public\\_License](https://es.wikipedia.org/wiki/GNU_General_Public_License) (**Última fecha de acceso:** 05/03/2017)
- [8] **Github:** Modificación sobre TabbedBoxMaker por Paul Hutchison **URL:** <https://github.com/paulh-rnd/TabbedBoxMaker> (**Última fecha de acceso:** 15/07/2017) - Versión original por Elliot White **URL:** <http://www.twot.eu/111000/111000.html> (**Última fecha de acceso:** 5/07/2017)
- [9] **Softeng:** Scrum **URL:** <https://www.softeng.es/es-es/empresa/metodologias-de-trabajo/metodologia-scrum/proceso-roles-de-scrum.html> (**Última fecha de acceso:** 20/07/2017)
- [10] **Wikipedia:** Madera DM **URL:** [https://es.wikipedia.org/wiki/Tablero\\_de\\_fibra\\_de\\_densidad\\_media](https://es.wikipedia.org/wiki/Tablero_de_fibra_de_densidad_media) (**Última fecha de acceso:** 05/03/2017)
- [11] **LXML:** Información sobre LXML **URL:** <http://lxml.de/api/lxml.etree.Element-class.html> (**Última fecha de acceso:** 10/08/2017)
- [12] **Imajeenyus:** Single line font **URL:** [http://imajeenyus.com/computer/20150110\\_single\\_line\\_fonts/index.shtml](http://imajeenyus.com/computer/20150110_single_line_fonts/index.shtml) (**Última fecha de acceso:** 20/07/2017)
- [13] **EvilMadScientist:** Hershey text **URL:** <http://www.evilmadscientist.com/2011/hershey-text-an-inkscape-extension-for-engraving-fonts/> (**Última fecha de acceso:** 02/06/2017)
- [14] **Myscrappinginspiration:** Single line font **URL:** <http://myscrappinginspiration.blogspot.com.es/2013/07/single-line-fonts-options.html> (**Última fecha de acceso:** 12/06/2017) - Descarga de la versión original **URL:** <https://www.dropbox.com/sh/yc5ff6l52wunean/WVYqEGZ9fF/paths2glyphs> (**Última fecha de acceso:** 10/07/2017)
- [15] **Instructables:** Láser bent wood **URL:** <http://www.instructables.com/id/Curved-laser-bent-wood/> (**Última fecha de acceso:** 01/08/2017)





