UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf

Escola Tècnica Superior d'Enginyeria Informàtica

Escola Tècnica Superior d'Enginyeria Informàtica

Universitat Politècnica de València

# Estudio de la progresión en el rendimiento de los algoritmos de rebanado 3D *open-source*

Trabajo Fin de Grado

**Grado en Ingeniería Informática**

**Autor**: Carlos Borrell Fortis

**Tutor**: Miguel Sánchez López

Curso 2020-2021

# Resumen

En este trabajo se expondrá qué es un programa de rebanado, cuál es su función y cómo ha progresado el rendimiento de estos programas desde su concepción hasta el día de hoy. Para ello se analizarán principalmente dos programas de rebanado 3D de código abierto *PrusaSlicer* y *Ultimaker Cura*; dos programas desarrollados por *Prusa3D* y *Ultimaker* respectivamente con política de código abierto que permite a los usuarios finales estudiar el código a fondo y proponer mejoras o cambios. Durante este proyecto realizaremos un recorrido por la historia de estos programas parándonos a estudiar las mejoras más sustanciales que han sucedido para así intentar predecir cómo avanzará esta tecnología en el futuro.

**Palabras clave:** progresión; rendimiento; rebanado; 3D; open-source; laminado

# Abstract

This paper will discuss what a slicing program is, what its function is, and how the performance of these programs has progressed from inception to the present day. For this, two open-source 3D slicing programs *PrusaSlicer* and *Ultimaker Cura* will be analyzed; two programs developed by *Prusa3D* and *Ultimaker* respectively with an open-source policy that allows end users to study the code in depth and propose improvements or changes. During this project we will take a tour through the history of these programs stopping to study the most substantial improvements that have occurred in order to try to predict how this technology will advance in the future.

**Keywords :** progression; performance; slicer; 3D; open-source; slicing

# Índice

# 1. Introducción

La impresión 3D es una tecnología que aún a día de hoy puede parecer ciencia ficción. Con cerca de 40 años de historia, la gente que no ha trabajado con ella desconoce de su funcionamiento e incluso existencia, y la gente que sí trabaja con ella, muchas veces desconocemos su funcionamiento interno.

Esta técnica de fabricación puede llegar a ser extremadamente compleja si se carece de los conocimientos necesarios de programación y mecánica. Por suerte para todos, la comunidad de personas que utilizan esta tecnología pone a disposición de todos sus conocimientos y experiencia. Gracias a esto hoy podemos tener en nuestro escritorio de casa o en la habitación de al lado una impresora 3D a la que le hayamos enviado un archivo y ella, poco a poco; nos va generando un modelo tridimensional físico.

En este proyecto veremos en qué consiste y cuál es la historia de la tecnología de fabricación aditiva. Además, estudiaremos qué papel tan fundamental tiene la informática en el desarrollo de la impresión 3D y cómo la comunidad puede aportar a este desarrollo a través de los proyectos *open-source* como los programas de laminado *PrusaSlicer* y *Ultimaker Cura*.

## 1.1. Motivación

En el grado superior de Ingeniería Informática de la UPV la tecnología de impresión 3D es invisible hasta el 4º curso de la carrera con dos asignaturas “Impresión 3D” y “Diseño y modelado 3D”. Para el momento en el que empecé estas asignaturas ya tenía una impresora 3D que utilizaba para imprimir miniaturas para juegos de rol. El tener esta tecnología despertó en mí una gran curiosidad y rápidamente consumí mucho contenido audiovisual sobre el tema y cuando supe que estas dos asignaturas estaban disponibles rápidamente me matriculé en ellas. Quería saber más de esta tecnología y cómo funcionaba hasta el punto de retrasar la entrega del TFG para poder cursar una de las dos ya que me coincidía con otra. Por supuesto, habiendo cursado otros tres años de la carrera y sabiendo que el TFG era algo obligatorio y tenía que estar relacionado con la carrera decidí combinar las dos cosas; lo que me gustaba y lo que había estudiado. De esta manera podía exponer mis conocimientos académicos y extraacadémicos.

## 1.2. Objetivos

Los objetivos que se pretenden lograr con este trabajo son los siguientes:

- Exponer en qué consiste la impresión 3D.
- Exponer su historia de manera que se comprendan sus orígenes.
- Informar del papel que tiene la informática en este campo de la tecnología describiendo dos programas de laminado.
- Analizar cómo estos programas llevan a cabo el proceso de laminado, su función más importante.
- Razonar qué estrategia es la más conveniente y eficaz.

## 1.3. Estructura del trabajo

La estructura que se seguirá en este trabajo es la siguiente:

En el segundo apartado “Fundamentos teóricos”, el lector podrá encontrar una explicación sobre qué es la impresión 3D, qué tipos de impresión 3D existen y cuál es la conexión entre esta tecnología y la informática a través de los programas de laminado. A continuación, en “Historia”, se expondrá la historia de la tecnología y de los programas de laminado de manera consecutiva. Seguidamente en “Estado del arte”, el trabajo cuenta con una explicación de cómo estos programas laminan las mallas de los objetos para imprimirlos. En “Comparativa” se considerarán los datos aportados en el apartado anterior y se ampliará sobre ellos para ver cuál es el coste temporal teórico de la funcionalidad anteriormente expuesta. Por último, en “Conclusiones” se procederá a analizar los resultados obtenidos y de manera lógica concluir si alguno de los dos programas es más eficaz que el otro.

# 2. Fundamentos teóricos

La impresión 3D (3 Dimensiones) es un campo de la tecnología que lleva activo y en desarrollo unos cuantos años. Sin embargo, a pesar de ello es una técnica de fabricación poco explorada por el público general lo cual lleva a pensar que es muy elitista y compleja. Por supuesto este no es el caso, la impresión 3D tiene diferentes partes que deben trabajar conjuntamente para producir un resultado de alta calidad. En este aspecto podría decirse que es compleja, sin embargo, hoy en día la producción de herramientas, tanto maquinaria como programas; para trabajar con esta tecnología se han desarrollado de manera centrada en un usuario final común y no especializado en la materia lo que permite que el proceso entero sea más sencillo. A continuación, se expondrán y explicarán las diferentes partes que conforman esta tecnología, introduciendo al lector en los conceptos necesarios para comprender en su totalidad su desarrollo y su funcionamiento.

## 2.1. La impresión 3D

La definición más aceptada y extendida de la impresión 3D es como sigue: “La impresión 3D es una tecnología o técnica de fabricación aditiva donde un objeto tridimensional es creado al superponer sucesivas capas de material.” [15] Eso significa, de manera simplificada; que la impresión 3D no es más que múltiples iteraciones de impresión 2D encima unas de otras.[2]

*Imagen 1. Caja negra impresa en 3D*

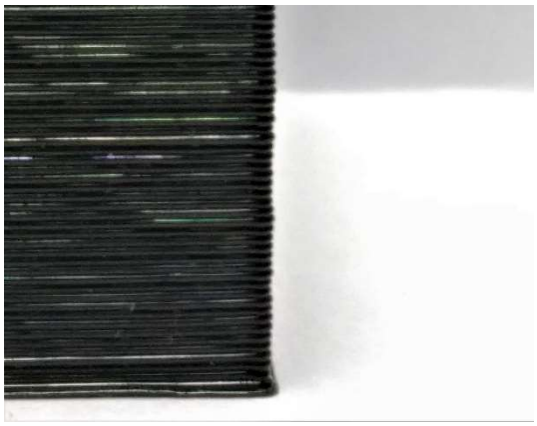

*Imagen 2. Detalle capas caja negra*

Esta tecnología que lleva cerca de 40 años de existencia permite crear objetos imposibles de fabricar con cualquier otro tipo de tecnología. La impresión 3D se desarrolla como técnica y herramienta de trabajo en muchos campos diferentes, desde la industria del automóvil hasta la creación de obras de arte pasando por el diseño de producto en donde tiene su mayor nicho de mercado y uso. Para generar estas estructuras tridimensionales existen diferentes técnicas de impresión 3D, diferentes tipos de máquinas, materiales, programas y configuraciones. El proceso creativo puede verse muy afectado por la finalidad del producto resultante, de todas formas, hay una serie de pasos o fases que suceden de normal hasta que el producto está totalmente finalizado.

Podemos dividir las fases de creación de producto en un total de cuatro: Diseño del producto, creación del modelo tridimensional, proceso de laminado o rebanado y por último la impresión. A continuación, se presentarán estas fases de manera completa y simple en el orden en el que se han presentado.

a) Diseño del producto

Al inicio de esta fase el producto es aún una idea, un concepto sin forma y sin determinar, el diseñador debe entender qué quiere plasmar en el objeto, el uso que se le va a dar una vez finalizado y cómo generar la mayor comodidad durante su utilización para el usuario final. Además, también se debe pensar en la fase de producción, la fase en la que el objeto se imprimirá. Esto último parece fácil, la impresión 3D tiene fama de ser muy versátil y así es, sin embargo; la cantidad de material que se debe utilizar para

realizar una impresión y el tiempo que requiere la máquina para llevarla a cabo son muy dependientes del diseño del producto entre otros factores. Un diseño con muchos detalles tardará más en completarse que un producto que requiere una menor resolución. En este aspecto hay diferentes características del diseño que se deberán tener en cuenta para optimizar la fabricación del producto. Entre ellas destacan, la cantidad de detalles, las partes flotantes y el número de objetos que conforman el producto final.

Los detalles de un objeto son cualquier tipo de relieve que salga o entre por alguna de las caras del objeto principal. Para generar estos relieves y que sean visibles, dependiendo del tamaño del detalle; es posible que tenga que reducirse la altura de capa a la que se imprime el objeto generando por tanto más capas para aumentar la resolución del producto final. Algunos ejemplos serían cualquier grabado en un objeto o pestañas y solapas que puedan servir para una mejor sujeción de este.

Las partes frontales son aquellas que se alejan del eje del producto y que no se apoyan en su parte inferior sino en el lateral o la parte superior. Algunos ejemplos de esto son las alas de un ángel o el enganche metálico de un bolígrafo estilográfico. Suponen un considerable aumento del tiempo y consumo de material porque para poder generar estas estructuras las piezas deben estar apoyadas en algo, esto significa que se deben generar estructuras de apoyo o soportes para poder empezar a hacer estas partes flotantes.

Por último, el número de objetos que componen el producto final influye en gran medida en el coste de impresión de manera proporcional, siendo que cuantos más objetos se deban imprimir, irremediablemente; más tardará en completarse la impresión del producto entero. Algún ejemplo es la comparación entre un llavero móvil y uno fijo o sólido.

b) Creación del modelo tridimensional

Un modelo tridimensional es una representación digital de un objeto ya sea real o imaginario formado por un conjunto de polígonos, comúnmente conocido como malla. Esta malla consiste en una lista de triángulos ya que cualquier superficie se puede dividir en polígonos, y los más pequeños, y por tanto más manejables; son los triángulos. Dado

esto, para generar un modelo tridimensional hay dos sendas posibles. Una de ellas es el escaneo del objeto del cual se pretende obtener el modelo y la otra senda consiste en crear digitalmente el modelo del objeto.

Para escanear un objeto, por supuesto este debe existir ya en una realidad tridimensional. El proceso consiste en la adquisición de gran cantidad de imágenes de referencia del objeto. Estas imágenes deberán representar el objeto desde diferentes ángulos de manera que, al introducirlas como datos de entrada en el programa correspondiente, este sepa interpretar correctamente las dimensiones del objeto y sus detalles.

Cuando hablamos de crear un objeto tridimensional de forma digital hablamos de modelado y podemos partir de objetos ya creados o empezar desde cero sin ningún tipo de referencia. Para crear objetos tridimensionales existen gran variedad de programas, cada uno dirigido a un tipo diferente de modelado. Gran variedad de programas, conociendo cómo se transcriben los modelos al archivo que los contiene ofrecen formas geométricas para modelar facilitando su transcripción, mientras que otros más refinados y avanzados permiten utilizar técnicas de modelado más orgánicas con las que se obtendrán resultados muy diferentes.

c) Proceso de laminado o rebanado.

Como hemos podido comprobar con la definición de impresión 3D, la estructura resultante se genera a partir de capas superpuestas de material. Para poder hacer esto es necesario utilizar un programa de laminado que partirá la malla del objeto en múltiples capas. Estos programas de laminado son múltiples y variados en diseño y accesibilidad. No obstante, su funcionalidad básica es constante entre todos ellos, y esta es generar código legible por la máquina que imprimirá el objeto. Este código se llama *g-code* y es el que pueden leer las impresoras 3D, consiste en una serie de instrucciones que desplazan el cabezal de la máquina y ajustan diferentes parámetros durante la impresión.

Los programas de laminado han evolucionado juntamente con la técnica de impresión 3D y son una parte fundamental del proceso. En este trabajo nos centraremos y ahondaremos en la función básica y conjunta de todos estos programas de laminar o

rebanar las mallas de los objetos para así generar las capas en las que se creará el producto final.

d) La impresión

La última fase del proceso de creación del producto es la impresión 3D del objeto. En esta fase convergen todas las decisiones tomadas durante las fases anteriores. Es además la fase menos controlada por el usuario ya que está totalmente dirigida y llevada a cabo por la máquina. Durante esta fase, la impresora leerá el archivo de tipo *g-code* que se le habrá proporcionado y llevará a cabo la impresión capa a capa hasta completar el objeto, momento en el que se podrá retirar de la base de impresión.

## 2.2. Tipos de impresión 3D

La impresión 3D es una tecnología con un alto grado de adaptabilidad y esto no es algo sencillo de conseguir para cualquier técnica creativa, diferentes materiales requieren de diferentes tratamientos, así como algunas formas requieren unos movimientos o una precisión que otras no requieren. En el campo de la impresión 3D esta adaptabilidad se consigue con diferentes máquinas y diferentes formas de llevar a cabo el proceso de creación de estructuras tridimensionales. Existen un total de seis tipos o técnicas de impresión 3D, estas son: extrusión de material fundido, fotopolimerización, fusión de material en polvo (polímeros), inyección de material, proyección de pegamento y fusión de material en polvo (metales). Cada una de ellas utiliza un principio diferente e impresoras adaptadas a ellos de manera que los resultados pueden variar drásticamente al utilizar una forma u otra. A continuación, se expondrán brevemente sus características individuales, así como su funcionamiento general. Téngase en cuenta de todas formas, que la tecnología de extrusión de material es la más extendida y la que utilizaremos como referencia en este trabajo.

- Extrusión de material fundido: Este tipo de fabricación es el más desarrollado para un consumidor no experto y es en el que nos centraremos en este trabajo. También conocido como modelado por deposición fundida o FDM, consiste en empujar un filamento de un material termoplástico a través de una boquilla

caliente lo que hace que este material se funda y caiga de manera constante por la boquilla. Esta a su vez está enganchada al cabezal de la impresora que se desplaza de acuerdo con las instrucciones dadas para generar capa a capa hacia arriba la estructura final. Una vez depositado el material este comienza a enfriarse y solidificarse de nuevo habilitando así una nueva superficie de deposición para la siguiente capa del objeto.[1]

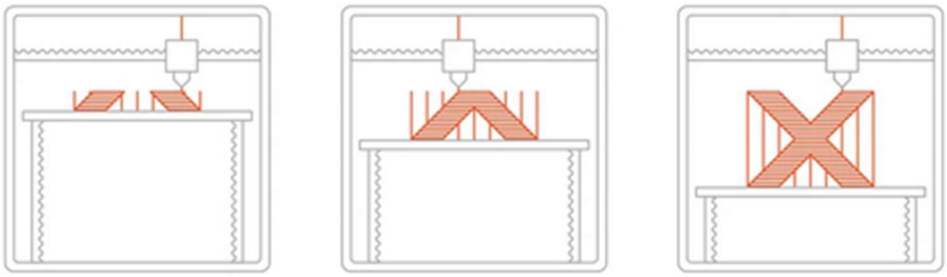

*Imagen 3. Esquema proceso de extrusión de material fundido*

| | |
|---|---|
| Precisión | Detalles de aproximadamente 0.5 mm |
| Aplicaciones comunes | Carcasas eléctricas, prototipado rápido, plantillas y fijaciones. |
| Ventajas | Buen acabado de la superficie, gran rango de materiales y colores. |
| Desventajas | Frágil, poca resistencia mecánica, coste elevado para resultados visuales. |

- Fotopolimerización: Este es un proceso de impresión 3D en el que se utiliza una resina almacenada en un tanque con fondo transparente de manera que, mediante la acción combinada de un láser y un espejo móvil, se curan de manera selectiva ciertas zonas de la base del tanque endureciendo la resina. Una vez la resina ha sido endurecida, se retrae el cabezal al que está enganchada dejando fluir otra capa de resina debajo y repitiendo el proceso hasta completar el objeto.[1]

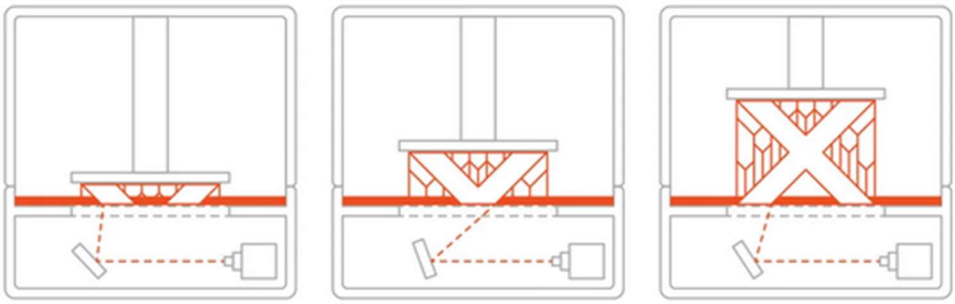

*Imagen 4. Esquema proceso de fotopolimerización*

| | |
|---|---|
| Precisión | Detalles de aproximadamente 0.15 mm |
| Aplicaciones comunes | Prototipos, joyería y aplicaciones médicas (dentales o audífonos) |
| Ventajas | Acabado liso, alta precisión |
| Desventajas | Frágil, escasa resistencia mecánica |

- Inyección de material: Esta tecnología funciona mediante la extrusión y posterior secado selectivo del material a través del cabezal de la impresora. El material solo se endurece en las zonas deseadas lo que facilita el proceso de acabado final. El material que se utiliza en esta tecnología son resinas fotopolímeras que se endurecen con la luz, el uso de un foco de luz con gran capacidad de enfoque es imperativo para generar un buen producto final.[1]

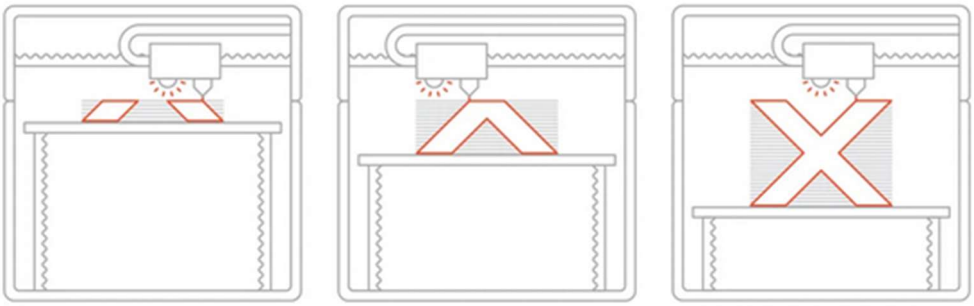

*Imagen 5. Esquema proceso de inyección de material*

| | |
|---|---|
| Precisión | Detalles de aproximadamente 0.1 mm |
| Aplicaciones comunes | Prototipos a color, moldes de inyección exclusivos, modelos médicos. |
| Ventajas | Gran acabado superficial, múltiples colores y materiales disponibles. |
| Desventajas | Frágil, poca resistencia mecánica, más costos que la fotopolimerización para fines visuales. |

- Fusión de material en polvo (polímeros y metales): Estos dos procesos a pesar de usar diferentes materiales tienen un funcionamiento muy similar, es por ello que se agruparán de esta manera. La fusión en lecho de polvo es un proceso en el que mediante una fuente de calor se funden selectivamente zonas de una capa de polvo del material a utilizar, ya sean metales o plásticos. La zona donde se llevará a cabo la impresión se irá desplazando hacia abajo conforme se completen las capas. Estas capas serán delgados depósitos del material a utilizar en forma de polvo de manera que al fundirse la zona particular esta se adhiere a la capa previa. El material se almacena en un depósito y mediante una pieza móvil se rellena la siguiente capa de la estructura repitiendo el proceso.[1]

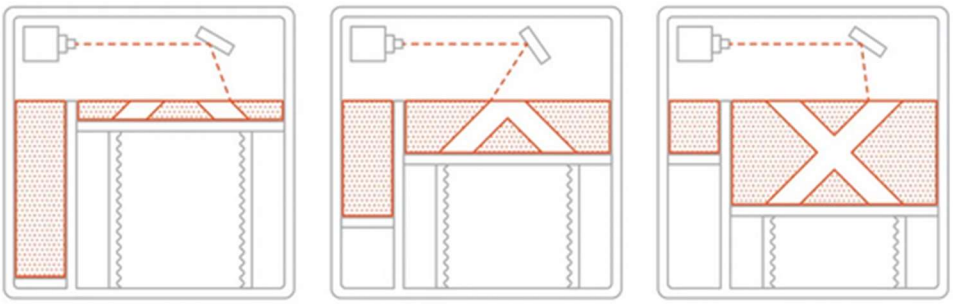

*Imagen 6. Esquema proceso de fusión de material en polvo*

- Fusión de material en polvo, polímeros:

| | |
|---|---|
| Precisión | Detalles de aproximadamente 0.3 mm |
| Aplicaciones comunes | Piezas funcionales, conductos complejos y producción de piezas exclusivas. |
| Ventajas | Piezas geométricamente complejas con buena resistencia mecánica |
| Desventajas | Requiere bastante tiempo y es más costosa que la impresión por extrusión de material |

- Fusión de material en polvo, metales:

| | |
|---|---|
| Precisión | Detalles de aproximadamente 0.1 mm |
| Aplicaciones comunes | Piezas funcionales, piezas médicas y dentales. |
| Ventajas | Piezas geométricamente complejas y robustas. |
| Desventajas | Precio elevado y tamaño pequeño. |

- Proyección de pegamento: La proyección de pegamento, o *binder jetting* es una técnica de impresión 3D en la que un pegamento une selectivamente regiones de una capa de polvo de material. En este aspecto es muy similar a la impresión en lecho de polvo con la diferencia que en vez de fundir zonas concretas las aglutina. Este aglutinante es depositado sobre el lecho de polvo mediante un cabezal que se mueve por encima de la plataforma de impresión. Al acabar el proceso se requiere un tiempo de endurecimiento para posteriormente retirar el polvo sobrante. [1]

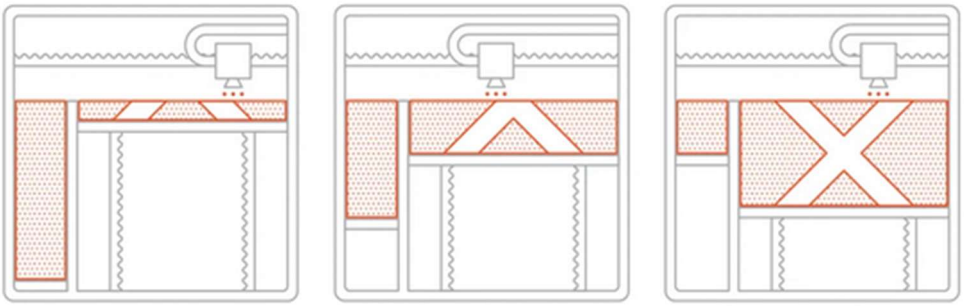

*Imagen 7. Esquema proceso de proyección de pegamento*

| | |
|---|---|
| Precisión | Detalles de aproximadamente 0.2 o 0.3 mm |
| Aplicaciones comunes | Prototipos a color, piezas metálicas funcionales o fundición de arena. |
| Ventajas | Bajo coste, gran plataforma de construcción, piezas metálicas funcionales. |
| Desventajas | Las propiedades metálicas no están a la altura de aquellas piezas hechas por fusión de material en polvo metálico. |

## 2.3. Programas de laminado

Como hemos mencionado en el apartado “c) Proceso de laminado o rebanado”, de las fases de la impresión 3D descritas en el punto “2.1. La impresión 3D” de este trabajo; las máquinas utilizadas para llevar a cabo el proceso de impresión 3D, tienen un lenguaje específico en el que les llegan las instrucciones necesarias para llevar a cabo esta tarea. Este lenguaje comúnmente se conoce como *G-code* o *gcode* y controla, en el caso de las impresoras; la temperatura del extrusor, la temperatura de la cama de calor, que es la plataforma de impresión; la velocidad de los motores, el movimiento de cabezal, la velocidad de alimentación del cabezal, así como su posición entre otros muchos parámetros. Es decir, el archivo leído por la máquina la controla en su totalidad para generar los resultados deseados. Ahora bien, estos archivos *gcode* son largos y muy específicos lo que dificulta en gran medida su manufacturación, por ello son generados utilizando programas de laminado o *slicers*. Estos programas son los encargados de rebanar digitalmente la malla por la que están formados los objetos tridimensionales, y de generar el código con las instrucciones necesarias para llevar a cabo una impresión 3D de acuerdo con las especificaciones del usuario.

Los programas de laminado constan de múltiples variables accesibles al usuario que puede modificarlas como crea conveniente. Por supuesto que no cualquier valor en estas variables es correcto o dará un buen resultado. Es por ello que en algunos de estos programas se han implementado niveles de conocimiento sobre el funcionamiento de la impresión 3D y los parámetros que se controlan. Estos niveles, conforme aumentan proporcionan mayor versatilidad a la par que riesgo, por ello un usuario primerizo solo podrá cambiar la altura de capa y el ángulo a partir del cual se generará material de soporte; mientras que un usuario experimentado podrá cambiar la densidad de los soportes, la temperatura del extrusor o la velocidad de alimentación del material entre otros. Todos estos parámetros y otros muchos más se tienen en cuenta por el programa a la hora de generar el *gcode* previamente mencionado.

En este trabajo nos centraremos en la sección de código de estos programas que genera las láminas o cortes en el objeto para observar cómo ha evolucionado e intentar predecir cómo cambiará en un futuro. Como la cantidad de programas de laminado es demasiado grande se concentrará este estudio en dos programas concretos, PrusaSlicer y Ultimaker Cura.

# 3. Historia

En este trabajo se pretende explorar la evolución del proceso de laminado de modelos tridimensionales llevado a cabo por los programas de laminado *PrusaSlicer* y *Ultimaker Cura*. Para ello primero es imprescindible conocer su origen y su concepción y desarrollo. Es por este motivo que en este apartado se expondrá a grandes rasgos los orígenes de la impresión 3D y más tarde los orígenes de las empresas creadoras de los programas a estudiar.

## 3.1. Orígenes de la impresión 3D

Los primeros indicios de una tecnología de fabricación aditiva datan del año 1981 cuando el Dr. Hideo Kodama, en Japón; propuso un sistema de prototipado rápido basado en el curado de resina. Sin embargo, a pesar de pedir la patente no pudo completar el proceso en el plazo establecido y por falta de financiación abandonó el proyecto. Algo después, en 1984 en Francia; fue un grupo de investigadores del CNRS (Centro Nacional de Investigación Científica) de Francia los que intentaron llevar a término una tecnología similar. Sin embargo, otra vez, la falta de financiación adecuada evitó que tramitaran la patente. [4]

Fue en 1986 cuando en Estados Unidos le fue concedida la patente, después de dos años de investigación y desarrollo; a Charles Hull, un fabricante de mesas y muebles. Hull decidió montar su propia empresa basada en este nuevo sistema de fabricación al que bautizó como estereolitografía. La empresa tiene por nombre “3D Systems” y en 1988 lanzó al mercado su primer producto comercial, la impresora de resina SLA-1. [3]

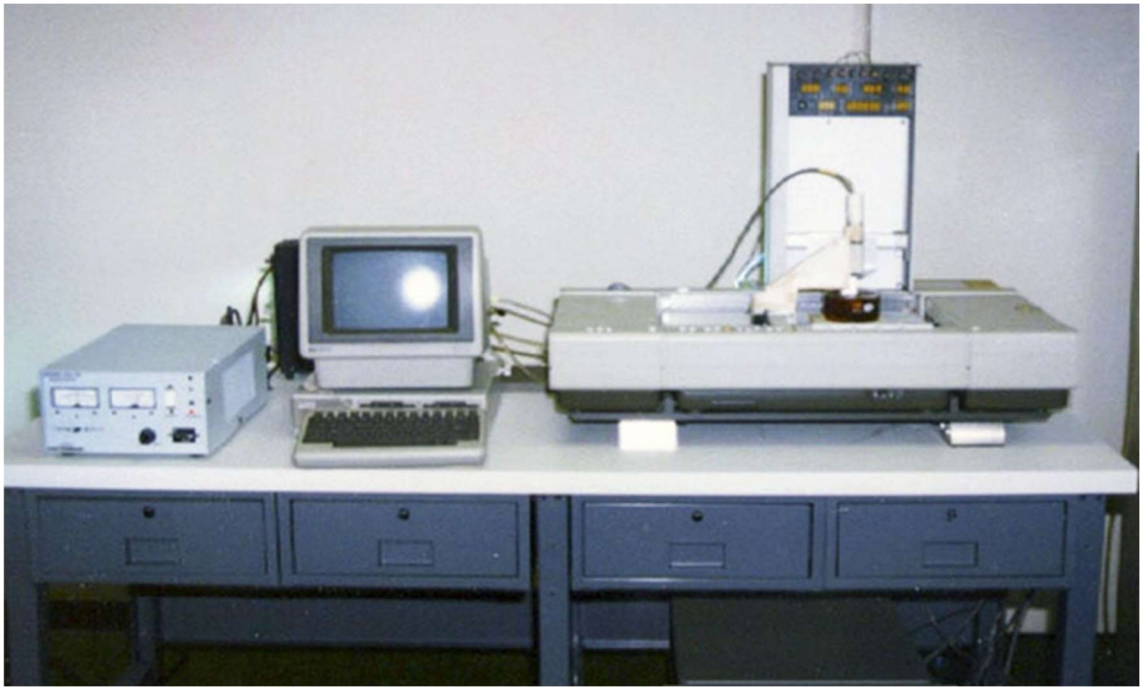

*Imagen 8. 3D SYSTEMS SLA-1*

En este mismo año se inventó otra técnica de impresión 3D, esta se basaba en fundir zonas concretas de una cama de polvo de plástico. La tecnología de tipo fundido de material en lecho de polvo fue registrada por Carl Deckard, universitario de Texas. La impresora que llevaba a cabo esta tarea se llamó Betsy, su calidad no era buena pero el concepto le dió otro impulso a la fabricación aditiva.

A la par que esta tecnología se aprobaba, Scott Crump, cofundador de Stratasys; desarrollaba la fabricación por fusión y deposición de material que a pesar de ser más simple que las anteriores llegó después. A esta tecnología se le concedió la patente en 1992 y rápidamente encontró aplicaciones en el campo médico.

No debemos olvidar que estas primeras máquinas eran enormes en comparación con las impresoras de sobremesa que se pueden obtener hoy en día. Además, su velocidad de impresión y calidad no tendrían cabida en el mundo actual, sin embargo, en aquel entonces, hace 30-40 años; el uso de una impresora que no requería supervisión humana y era capaz de generar prototipos de relativa fidelidad supuso un impulso en la industria del cual algunas empresas se beneficiaron. Es en 1993 cuando otra empresa importante entonces, menos conocida ahora; llamada ZCorp desarrolló una tecnología basada en las impresoras de tinta comunes. Esta máquina abrió el camino a la tecnología que ahora se conoce como proyección de pegamento. [11]

"En menos de diez años, la impresión 3D había pasado de ser una idea elocuente sobre un papel a una opción viable en fabricación a pequeña escala." [11] Las impresoras fueron mejorando junto con el avance tecnológico reduciendo su tamaño y coste y aumentando su potencia, velocidad y características. A principios del siglo XXI las impresoras adoptan múltiples colores de materiales y en 2005 comienza el movimiento RepRap. El movimiento RepRap (*Replicating Rapid-Prototypers*) es un proyecto iniciado por el Dr. Adrian Bowyer que tenía como idea principal conseguir un objetivo simple pero original e innovador, hacer que las impresoras 3D se imprimieran a sí mismas. Por supuesto esto planeaba una serie de problemas como que el tamaño de la impresora no podía ser más grande que la impresora original y así sucesivamente. Sabiendo las limitaciones que tendría este proyecto se decidió imprimir partes de la impresora por separado para luego poder montarla una vez se tuvieran todas las partes. Esta idea, hecha por una comunidad y sin una patente restrictiva; rápidamente recibió el apoyo de muchos seguidores alrededor del mundo. En 2008 el proyecto da su primer avance claro al crear la primera impresora 3D capaz de replicar una gran cantidad de sus propias piezas, la llamaron "Darwin". [3] A partir de ese momento el coste de producción de nuevas máquinas baja drásticamente y empiezan a venderse paquetes de montaje propio o *DIY kits* que contienen todas las piezas requeridas para montar la impresora y es el usuario final el que la monta. Trasladando el coste de la mano de obra al tiempo del usuario y reduciendo el coste de los materiales necesarios para hacer la impresora, es entre 2009 y 2014 cuando muchas de las patentes relacionadas con la técnica de fabricación de impresión 3D dejan de tener vigencia. Este suceso junto con el desarrollo de impresoras autorreplicables hace que la tecnología salte al mercado mundial y se presente al mundo como una herramienta asequible y potente, capaz de revolucionar las industrias.

A la par que se evolucionaba en el hardware, la comunidad lo hacía también y crecía exponencialmente. Es en 2008 cuando una pequeña página web llamada "Thingiverse" empieza a dejar a sus usuarios compartir en red sus diseños y modelos 3D para que otros puedan utilizarlos en sus impresoras de manera gratuita. Rápidamente la página

ganó afluencia y esto permitió un rápido desarrollo en el diseño de piezas para impresoras u objetos decorativos o prácticos. [11]

El diseño tridimensional empieza a tomar relevancia en muchas industrias, en especial en el campo de la salud donde las impresoras 3D, sobre todo las que trabajan con resinas se empiezan a utilizar para crear prótesis personalizadas, resistentes y asequibles. Es en este campo donde la impresión 3D cobra mucha fuerza y continúa desarrollando su alcance con tejidos orgánicos y funcionales hasta el día de hoy.

*Imagen 9. Pierna prostética impresa en 3D*

En cuanto a las impresoras de metales, estas comienzan a usarse con metales preciosos y aleaciones, creando joyas con diseños complejos, demasiado difíciles como para fabricarlas en masa. [3]

Hoy en día las impresoras 3D están al alcance de muchos por lo que su fabricación está mucho más centrada en el usuario final; que no tiene que ser un experto para obtener buenos resultados con sus máquinas. Las impresoras tienen precios que varían desde los 100$≈84€ hasta los 2.5M$≈2.1M€, capaces de imprimir desde válvulas para el corazón hasta casas enteras. Por supuesto no todas las impresoras sirven para todo y es por ello que el coste y los resultados pueden variar tan drásticamente. Lo que es claro es, que el avance de esta tecnología en sus 35 años de existencia es digno de reconocimiento y continúa favoreciendo la calidad y el precio. [6]

*Imagen 10. Casa impresa en 3D*

## 3.2. Los programas de laminado

En el apartado anterior se ha expuesto cómo ha sucedido el avance de la tecnología de impresión 3D. En este se expondrá la historia de los dos programas de laminado en los que se centrarán los siguientes apartados y el núcleo del trabajo. Para llevar a cabo esta tarea es irremediable hablar sobre las empresas detrás de estos programas; de cómo surgieron, su relación con el mundo de la tecnología de fabricación aditiva 3D y porqué son importantes en el campo del software que se explicará.

Primero que nada, es importante mencionar el origen de los programas de laminado con *Skeinforge*. Este fue un programa, ya obsoleto; escrito en python, que transformaba modelos tridimensionales en archivos de código *gcode* para impresoras 3D. [9] Se utilizó durante los inicios del movimiento *RepRap* y fue el precursor de este tipo de programas. De *Skeinforge* surgirían más tarde *Slic3r*, precursor de *PrusaSlicer*; y *Replicartor-G*, precursor de *Ultimaker Cura*. A continuación, se expondrá la historia de los programas y sus empresas, *PrusaSlicer*, de *Prusa 3D*; y *Ultimaker Cura*, de *Ultimaker*.

*Imagen 11. Logo PrusaSlicer*

*Imagen 12. Logo Ultimaker Cura*

- ***PrusaSlicer:*** PrusaSlicer es el *software* de laminado de la empresa *Prusa3D*. Esta empresa fundada y dirigida por el checo, Josef Prusa; en 2012, empezó en un sótano de Praga vendiendo su *Original Prusa I3* utilizando cajas de pizza para empaquetarlas. A pesar de estos humildes comienzos, la empresa creció rápidamente y adoptó el uso de un programa de laminado propio compatible y centrado en sus impresoras. [7] Esto les llevó a, en noviembre de 2016; decantarse por utilizar el proyecto de código abierto *Slic3r* como base para su nuevo programa, *Slic3r Prusa Edition* o *Slic3r PE*. *Slic3r* es un programa comunitario y de código abierto creado por Alessandro Ranellucci en 2011 como

una herramienta para generar código comprensible para las impresoras 3D. [10] Cuando *Prusa3D* decidió generar una rama propia, *Slic3r PE* continuó siendo de código abierto y totalmente gratuito pero respaldado y enfocado en la empresa y sus impresoras así como su comunidad. Durante el tiempo que se estuvo utilizando este programa, la comunidad lo utilizó indistintamente para las impresoras Prusa o para otras diferentes ya que el código resultante servía también en otras máquinas. Es por esto que cuando los problemas surgían era difícil determinar al responsable ya que las funcionalidades del programa base y de la edición de Prusa eran intercambiables y en muchos casos así había sido. En vista del gran problema al que se podrían tener que enfrentar, en *Prusa 3D* se decidió modificar el programa de tal manera que estas posibles disputas no fueran algo por lo que preocuparse. Tomada esta decisión, en mayo de 2019 *Prusa 3D* lanza *PrusaSlicer 2.0*, un programa con ciertos parecidos a *Slic3r PE* pero con muchas mejoras. Los parecidos más importantes se encontraban en el núcleo del programa, ya que se seguía basando en *Slic3r* y esto lo hacía de código abierto y gratuito. Además, la interfaz de usuario, a pesar de estar renovada; seguía siendo muy similar. En cuanto a los cambios, entre otros, el más importante a nivel empresarial fue el cambio de licencia que pasó a ser una licencia de empresa. En el campo técnico incluyeron las impresoras de resina como opción de laminado y aumentaron el número de opciones disponibles, así como modos de impresión y pequeñas mejoras de comodidad de uso. Hoy en día *PrusaSlicer* es uno de los programas líder en avances técnicos del mundo con el que se consiguen resultados de alta calidad. [5]

- ***Ultimaker Cura:*** Programa gratuito y de código abierto creado por David Braam quien comenzó poco después a trabajar para Ultimaker para poder mantener el programa. La historia de esta empresa comienza en Países Bajos en mayo del año 2011, cuando Martijn Elserman, Erik de Bruijn y Siert Wijnia se unieron con el objetivo de replicar la impresora 3D *Darwin*, la primera impresora autorreplicable. Esto no fue posible, pero en el proceso desarrollaron su primera impresora propia, la *Ultimaker Protobox*; que no es autoreplicable y tenía como nicho de mercado la industria del automóvil, arquitectura, educación o el campo de la salud. [12] Como en el caso de muchas empresas de impresión 3D, al principio el *software* de la empresa estaba basado en otro proyecto de código

abierto de la comunidad. En este caso el programa estaba basado en una versión de *Replicator-G* hasta el año 2016 cuando se adoptó *Cura* como el estándar para la empresa y se le dio respaldo total a la plataforma. *Cura*, como otros muchos programas de laminado; tiene ciertas licencias que permiten su uso a terceros, pero siempre dan la propiedad a *Ultimaker* que es quien decide que el programa sea de código abierto, un proyecto comunitario y gratuito. Cura ha sido transformado en cinco ocasiones diferentes con grandes actualizaciones que afectaron a la interfaz y las funcionalidades presentadas. Estas grandes transformaciones corresponden a las versiones: 2.3 en la que se añadieron algunas características así como un aumento en la velocidad de procesamiento; 3.0 en la que se integró el uso de CAD y *plugins*; 3.5 en la que se adoptó el uso de archivos *3MF* como formato estándar en vez del *STL;* 3.6 en la que se introducen perfiles de características de materiales para la impresión; y por último la 4.0 en la que se mejoró la interfaz de usuario así como se habilitó la evaluación de los *plugins* por parte de la comunidad. [14] Hoy en día *Ultimaker Cura* es responsable de la fabricación de más de un millón y medio de impresiones a la semana y cuenta con más de cien perfiles de impresoras compatibles.

# 4. Estado del arte

En este apartado se procederá a la exposición del código a estudiar en ambos programas, *PrusaSlicer* y *Ultimaker Cura*; en su versión más reciente. Nos centraremos en el funcionamiento de la parte del código encargada de generar las capas del modelo tridimensional.

## *4.1. PrusaSlicer*

La sección de código que se mostrará a continuación está escrita en el lenguaje de programación *C++*. Esto se debe a que durante la transición de *Slic3r PE* a *PrusaSlicer* la empresa *Prusa3D* decidió mantener el lenguaje *C++* para estandarizar las funciones básicas y primarias, mientras, lo referente a la interfaz y de cara al usuario, se utiliza *Perl* para darle un aspecto más moderno al programa.

```
template <Axis A>
void
TriangleMeshSlicer<A>::slice(const std::vector<float> &z, std::vector<Polygons>*
layers)
    const{
    std::vector<IntersectionLines> lines(z.size());
    {
        boost::mutex lines_mutex;
        parallelize<int>(
            0,
            this->mesh->stl.stats.number_of_facets-1,
            boost::bind(&TriangleMeshSlicer<A>::_slice_do, this, _1, &lines,
&lines_mutex,
                z)
        );
    }

    layers->resize(z.size());
    parallelize<size_t>(
        0,
        lines.size()-1,
        boost::bind(&TriangleMeshSlicer<A>::_make_loops_do, this, _1, &lines, layers)
    );
}
```

[8]

Esta sección de código corresponde a la parte central y primaria del programa. Esta no ha sido cambiada durante la transición mencionada previamente. Es por ello que no

encontraremos esta sección de código en el repositorio de *GitHub* de *PrusaSlicer,* sino que lo encontraremos en el de *Slic3r*.

Esta sección de código representa una parte del proceso de laminado en la que se toma un vector de coordenadas, que representa las alturas a las que se creará cada capa del objeto; y devuelve un vector de polígonos representando las capas generadas. Para esto *Prusa3D* y *Slic3r* utilizan librerías propias que mejoran el rendimiento del código, reduciendo el uso de memoria entre otras mejoras. Es por ello que se pasarán por alto las explicaciones de algunas funciones, o algunos elementos, cuya definición estará en alguna sección de código o en algún fichero del repositorio que no se ha podido estudiar.

Para empezar con el análisis del código se debe considerar el encabezado lo primero. Aquí se observa que lo que realmente se está escribiendo es una plantilla como especifica la palabra `template`. Una plantilla que podremos utilizar con todos los tipos de ejes. La definición de estos ejes se encuentra en otro fichero que no se considera necesario mostrar aquí. El uso de una plantilla en vez de una función estática se debe a que el mismo programa se puede usar para todas las impresoras Prusa independientemente del eje de referencia que utilicen. A continuación vemos que la función a la que se llama se encuentra en una de estas librerías propias de la empresa llamada `TriangleMeshSlicer` y la función `slice()` que recibirá como datos de entrada, el vector z con las alturas de corte y un vector de polígonos llamado `layers`. Este último vector será el vector que acabará con los polígonos formados por el contorno del objeto cuando el método acabe su ejecución.

Continuando con el proceso, primero se genera un vector que contiene las intersecciones de los planos, a las alturas designadas en el vector z; con los triángulos que conforman la malla del objeto. Inmediatamente después, se calculan de manera paralela estas intersecciones y se unen de manera que se forme el polígono que corresponda a la capa en la que se encuentra el proceso. Esto se lleva a cabo mediante la función `bind()`. Esta función recoge las intersecciones de los diferentes hilos de ejecución y las ubica comparándolas con la malla original y reformulando el polígono de manera que este sea paralelo a la base de laminado e impresión.

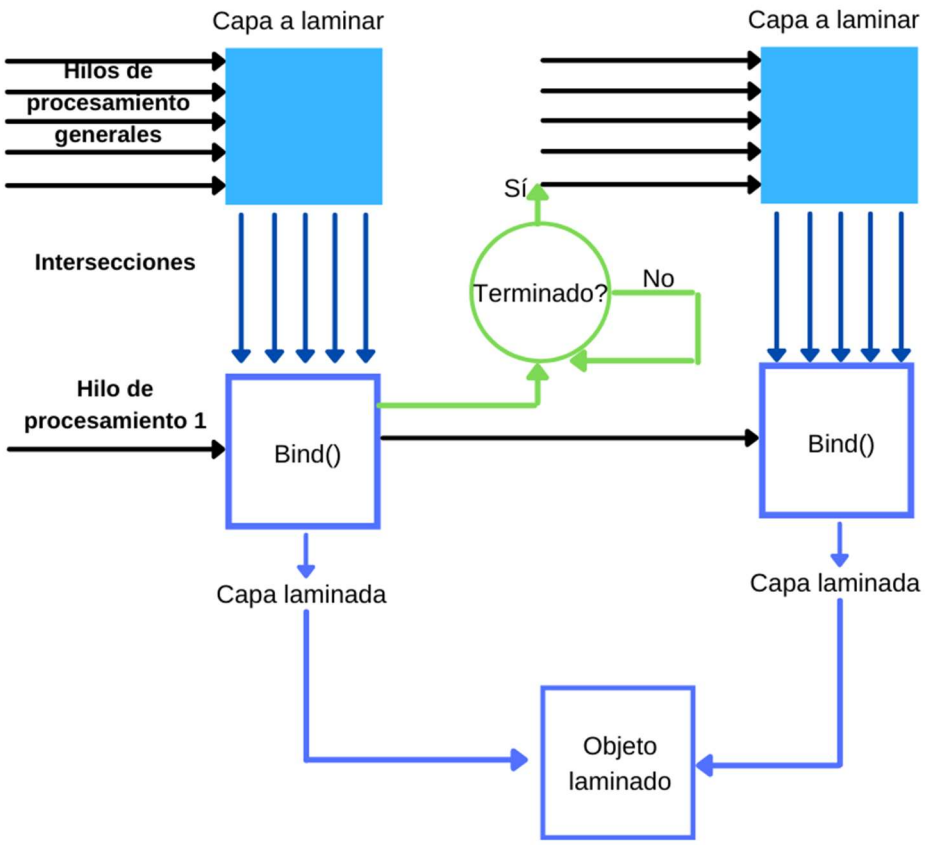


*Imagen 13. Diagrama laminado PrusaSlicer*

Seguidamente se expone la función `z.resize()` que escala el vector z para liberar espacio y repetir la ejecución de la función anterior. De esta manera capa a capa y utilizando múltiples hilos de procesamiento se consiguen un número de polígonos de capa igual al número de capas en las que se ha dividido el objeto. Cabe remarcar que estos polígonos solo se rigen por la regla de tener aristas y vértices, pero no tienen por qué ser regulares o tener un número determinado de aristas o vértices.

## 4.2. *Ultimaker Cura*

A continuación, como se ha hecho en el punto anterior se expondrá el código literal utilizado en el programa *Ultimaker Cura* para laminar las mallas de los objetos a imprimir. Después se explicará su funcionamiento y qué partes intervienen en el proceso de laminado.

```
void Slicer::buildSegments(const Mesh& mesh, const std::vector<std::pair<int32_t,
int32_t>>
    &zbbox, std::vector<SlicerLayer>& layers){
    // OpenMP
#pragma omp parallel for default(none) shared(mesh, zbbox, layers)
    // Use a signed type for the loop counter so MSVC compiles (because it uses OpenMP
2.0, an old version).
    for (int layer_nr = 0; layer_nr < static_cast<int>(layers.size()); layer_nr++){
        int32_t z = layers[layer_nr].z;
        layers[layer_nr].segments.reserve(100);
        // loop over all mesh faces
        for (unsigned int mesh_idx = 0; mesh_idx < mesh.faces.size(); mesh_idx++){
            if ((z < zbbox[mesh_idx].first) || (z > zbbox[mesh_idx].second))
                continue;
            // get all vertices per face
            const MeshFace& face = mesh.faces[mesh_idx];
            const MeshVertex& v0 = mesh.vertices[face.vertex_index[0]];
            const MeshVertex& v1 = mesh.vertices[face.vertex_index[1]];
            const MeshVertex& v2 = mesh.vertices[face.vertex_index[2]];
            // get all vertices represented as 3D point
            Point3 p0 = v0.p;
            Point3 p1 = v1.p;
            Point3 p2 = v2.p;
            SlicerSegment s;
            s.endVertex = nullptr;
            int end_edge_idx = -1;
            if (p0.z < z && p1.z > z && p2.z > z)              //  1_______2
            {                                                 //    \     /
                s = project2D(p0, p2, p1, z);                 //------------- z
                end_edge_idx = 0;                             //     \ /
            }                                                 //      0

            else if (p0.z > z && p1.z <= z && p2.z <= z)      //      0
            {                                                 //     / \      .
                s = project2D(p0, p1, p2, z);                 //------------- z
                end_edge_idx = 2;                             //   /     \    .
                if (p2.z == z)                                //  1_______2
                {
                    s.endVertex = &v2;
                }
            }

            else if (p1.z < z && p0.z > z && p2.z > z)         //  0_______2
            {                                                 //    \     /
                s = project2D(p1, p0, p2, z);                 //------------- z
                end_edge_idx = 1;                             //     \ /
            }                                                 //      1

            else if (p1.z > z && p0.z <= z && p2.z <= z)       //      1
            {                                                 //     / \      .
                s = project2D(p1, p2, p0, z);                 //------------- z
                end_edge_idx = 0;                             //   /     \    .
                if (p0.z == z)                                //  0_______2
                {
                    s.endVertex = &v0;
                }
            }
```

```
            else if (p2.z < z && p1.z > z && p0.z > z)          //  0_______1
            {                                                  //   \     /
                s = project2D(p2, p1, p0, z);                  //-------------- z
                end_edge_idx = 2;                              //     \ /
            }                                                  //      2

            else if (p2.z > z && p1.z <= z && p0.z <= z)        //      2
            {                                                  //     / \      .
                s = project2D(p2, p0, p1, z);                  //-------------- z
                end_edge_idx = 1;                              //   /     \    .
                if (p1.z == z)                                 //  0_______1
                {
                    s.endVertex = &v1;
                }
            }
            else{    continue;    }
            // store the segments per layer
            layers[layer_nr].face_idx_to_segment_idx.insert(
            std::make_pair(mesh_idx, layers[layer_nr].segments.size())
            );
            s.faceIndex = mesh_idx;
            s.endOtherFaceIdx = face.connected_face_index[end_edge_idx];
            s.addedToPolygon = false;
            layers[layer_nr].segments.push_back(s);
        }
    }
}
```

[13]

Como se puede observar esta sección de código es mucho más extensa que la expuesta en el apartado anterior ya que en vez de llamar a diferentes métodos, se ejecutan las instrucciones en un solo método. Esto es una mejora a nivel de lectura para el desarrollador, puesto que no necesita tener diferentes archivos abiertos para entender lo que está sucediendo. Además, de manera más acusada, es una mejora en términos de computación puesto que el programa no tendrá que hacer esa tarea de búsqueda de métodos en otros archivos o partes del código; cuando acabe la ejecución de una instrucción podrá continuar con la siguiente sin mayor complicación.

En cuanto a la ejecución del programa, lo primero que se encuentra son las tres variables que necesita el método para ejecutarse. `mesh` es el objeto de tipo malla que se va a laminar. Cada objeto que queramos imprimir, aunque sea simultáneamente, es una malla diferente. Después está `zbbox` que es un vector de parejas de números enteros de 32 bits de tamaño que se ha preparado en la función previa, y contiene en cada pareja la altura más alta y la más baja de cada triángulo de la malla. Esto servirá

más adelante en el método. Por último, el parámetro `layers` que contiene las capas o planos a las alturas que corresponde, pero sin intersecciones con el modelo.

Una vez llamado el método con los parámetros correspondientes se preparan los hilos de ejecución utilizando *OpenMP*, una interfaz que permite la computación paralela en diferentes hilos de procesamiento. A estos hilos se les da acceso compartido a los parámetros descritos anteriormente y se reparten en bucle las diferentes capas a procesar, de manera que, si uno de los hilos de procesamiento termina una capa antes que otros, puede pasar a la siguiente y hacer más en general. Una vez dentro de este bucle general se recorren todos los triángulos que se encuentran en la altura de la capa que se está procesando y se proyectan sobre esta. Las proyecciones se llevan a cabo en una serie de condiciones de tipo `if - else if` en las que se comprueba qué vértices son los que están por encima, y cuáles por debajo, ya que, dependiendo de la posición del triángulo con respecto a la capa, pueden darse tres casos con un comportamiento diferente cada uno.

El primer caso es que el triángulo esté incluido entero en la capa. En ese caso se omite el triángulo porque tendrá adyacentes otros tres que pueden salirse de la misma.

El segundo caso es el estándar y a su vez tiene dos opciones. Consiste en que dos vértices están en la capa, pero el tercero está por encima o por debajo. En caso que esté por encima se debe tener en cuenta y unir a la capa. Si está por debajo se descarta ya que, si se tuviera en cuenta en una capa, en la siguiente se debería descartar y esto supondría una ralentización del procesamiento, que tampoco aporta una mejora sustancial en la calidad de la impresión.

El último caso es aquel en el que el triángulo solo toca con un vértice el plano. En este caso también se desestima el vértice, pues con solo un punto no se pueden generar segmentos. Además, este vértice tendrá conexiones con otros triángulos, lo que probablemente indique que estos otros triángulos sí se encuentren en el segundo caso.

Como última parte del método dentro de cada capa, se guardan en la variable `layers` los segmentos generados, que unen todos los vértices de la malla que han intersecado con el plano. De esta manera termina el proceso de una capa y se busca la siguiente capa a procesar.

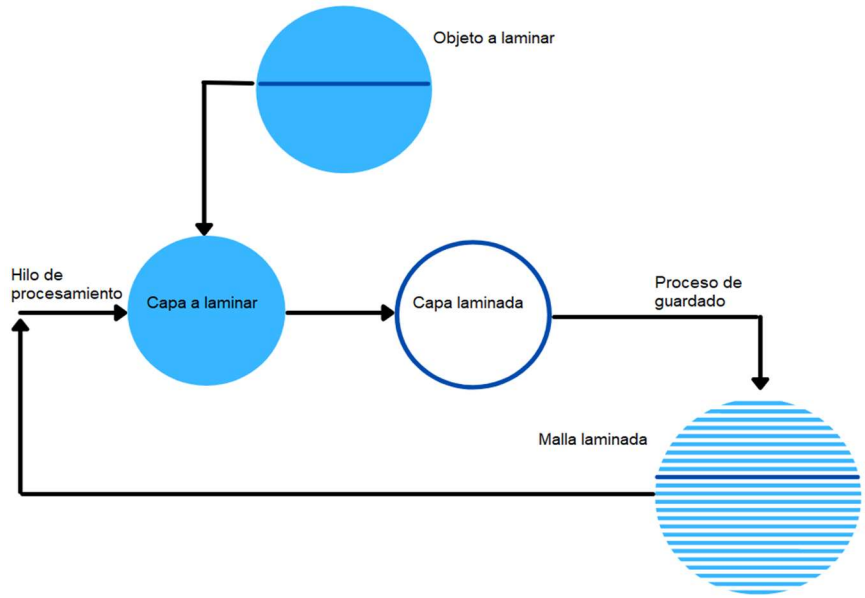


*Imagen 14. Diagrama de laminado Ultimaker Cura, 1 hilo*

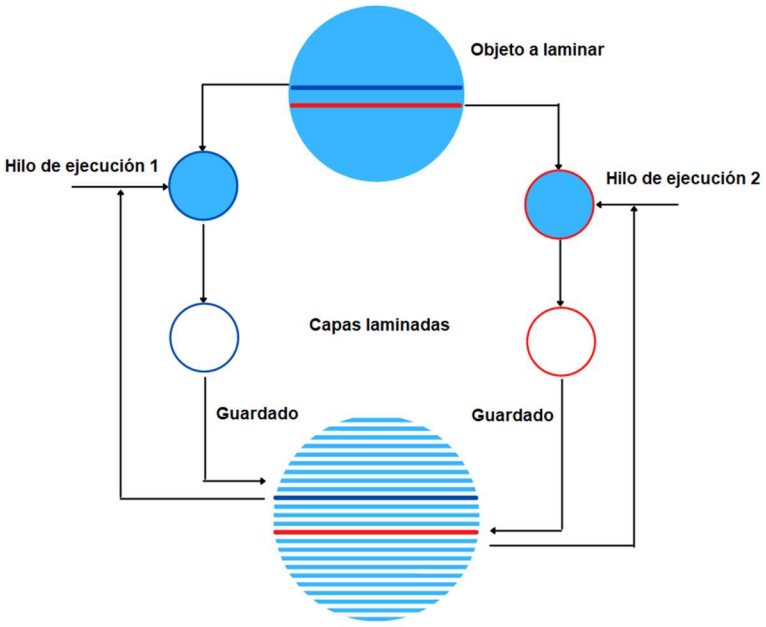


*Imagen 15. Diagrama de laminado Ultimaker Cura, 2 hilos*

# 5. Comparativa

Después de haber analizado y explicado el funcionamiento de las secciones de código en el apartado anterior, se procederá a explicar de manera rápida cuáles son los limitadores de velocidad de procesamiento de los métodos de laminado y compararlos entre sí. De esta manera debería ser posible comprender qué programa es más eficaz a la hora de laminar una malla. Para estas comparaciones solo se considerará la presencia de una malla a laminar y no de varias. En el caso de haber varias se debería estudiar qué interacción existe entre ellas, si se laminan una a una o se generan diferentes hilos de procesamiento para cada una. Hacer esta valoración aumentaría significativamente el grado de dificultad de realizar la valoración de eficacia y por tanto se dejará de lado en este trabajo.

En cuanto a los limitantes de velocidad del programa *PrusaSlicer* en el contexto del laminado de una malla, es necesario mencionar cuál es el comportamiento del método de manera resumida. La idea de esta función es la siguiente, primero se genera el número de capas en el que se deberá dividir la malla, después, capa por capa; se ejecutan diferentes hilos de procesamiento que encuentran las intersecciones entre el modelo y el plano de la capa que se está procesando en ese momento. Por último, un último hilo recoge esas intersecciones y las une entre sí generando polígonos y por tanto la capa laminada final. Con este método el procesamiento de cada capa está limitado a la velocidad de procesamiento de este último hilo puesto que hasta que no termine este no se puede avanzar a la siguiente capa. Este proceso se tendrá que repetir en todas las capas hasta el final del laminado de toda la malla. Por tanto, se podría decir que el tiempo que tarda el programa en finalizar el laminado es igual al número de intersecciones por el número de capas. En este caso se asume que el número de intersecciones es constante porque podría incrementar en algunas capas y reducirse en otras. Pero para no añadir más complejidad a la situación se asumirá un número constante de intersecciones durante todas las capas. Sabiendo esto se considera “i” como el número de intersecciones en una capa, “c” el número de capas y “t” como el tiempo necesario para unir una intersección al resto para formar el polígono final. La ecuación más sencilla sería $t * i * c = T$ considerando “T” como el tiempo total de procesamiento. En esta ecuación “t” es una variable dependiente de la máquina y las otras variables pueden diferir de modelo a modelo. Suponiendo una precisión infinita en la horizontal del modelo y una precisión infinita en la vertical del modelo; se sustituiría

“i” y “c” por infinito dejando una función de tipo cuadrática. En este modelo la variable que más influye en el cómputo de tiempo total de ejecución del método es el número de intersecciones, ya que, como se ha mencionado anteriormente; es el que más se repite; concretamente en cada capa y está totalmente determinado por “t”. El hilo de procesamiento encargado de conectar y crear cada polígono de cada capa es el cuello de botella en este programa, si de escalas muy grandes se habla. Por supuesto toda esta situación está influenciada por la máquina en la que se ejecuta el método, el número de intersecciones que hay por capa en el modelo y el número total de capas de las que consta dicho modelo. Como se muestra en la ecuación, cualquiera de estas variables afectará directamente al tiempo total de cómputo.

Por otra parte, en el caso del método utilizado por *Ultimaker Cura* el método funciona de la siguiente manera, primero se asigna cada hilo de procesamiento a una de las capas que se han preparado previamente. Después estos hilos comienzan a seleccionar las intersecciones del modelo con las capas y a juntarlas en su polígono correspondiente. Una vez un hilo acaba con su capa pasa a la siguiente disponible. De esta manera, si se diera el caso que un hilo trabaja en capas más pequeñas, haría más capas que el resto de hilos. Sabiendo esto, se puede extrapolar que, aunque el número de capas pueda variar drásticamente entre hilos, el número de intersecciones se mantendrá constante en su mayoría. Esto se debe a que el tiempo que se pueda tardar en copiar los polígonos resultantes al vector `Layers`, donde se almacena el resultado final de la malla laminada; puede ser negligible. En tal caso asumiendo de nuevo “t” como el tiempo que tarda un hilo en copiar el resultado, “i” como el tiempo que tarda en unir una intersección al polígono que está generando, y por último “c” como el número de capas en las que trabaje dicho hilo. Si de un solo hilo se tratase la función resultante sería exactamente igual a la propuesta con *PrusaSlicer* $t * i * c = T$ en la que “T” es el tiempo total de cómputo. Sin embargo, como en este caso los hilos son totalmente independientes entre sí y no requieren de otro hilo que recoja las intersecciones, es posible dividir el número de capas entre el número de hilos “h” de manera que la función resultante sea $t * i * ( c / h ) = T$ . Esta función supone que el número de capas se divide equitativamente entre todos los hilos para reducir su grado de complejidad.

A continuación, se expondrá un ejemplo de laminado de un objeto utilizando el programa *Ultimaker Cura*. No se utilizará *PrusaSlicer* debido a la ausencia de un resultado temporal a la hora de laminar.

Para este ejemplo el objeto a laminar será un cubo de lado 10 cm de manera que todas las capas sean aproximadamente del mismo tamaño como en el caso hipotético presentado arriba. La altura de capa general será de 0’2 mm, la altura de la primera capa será de 0’4 mm para conseguir un mejor agarre con la base de impresión. Sabiendo esto, el recuento de capas es de 499. La máquina sobre la que se ejecutará consta de dos núcleos capaces de soportar 4 hilos.

En las imágenes siguientes se puede observar el tamaño y algunos de los parámetros de la impresión, como la altura de capa en la parte superior derecha. A su lado veremos también el porcentaje de relleno que se utilizará. En este caso no afecta al resultado final.

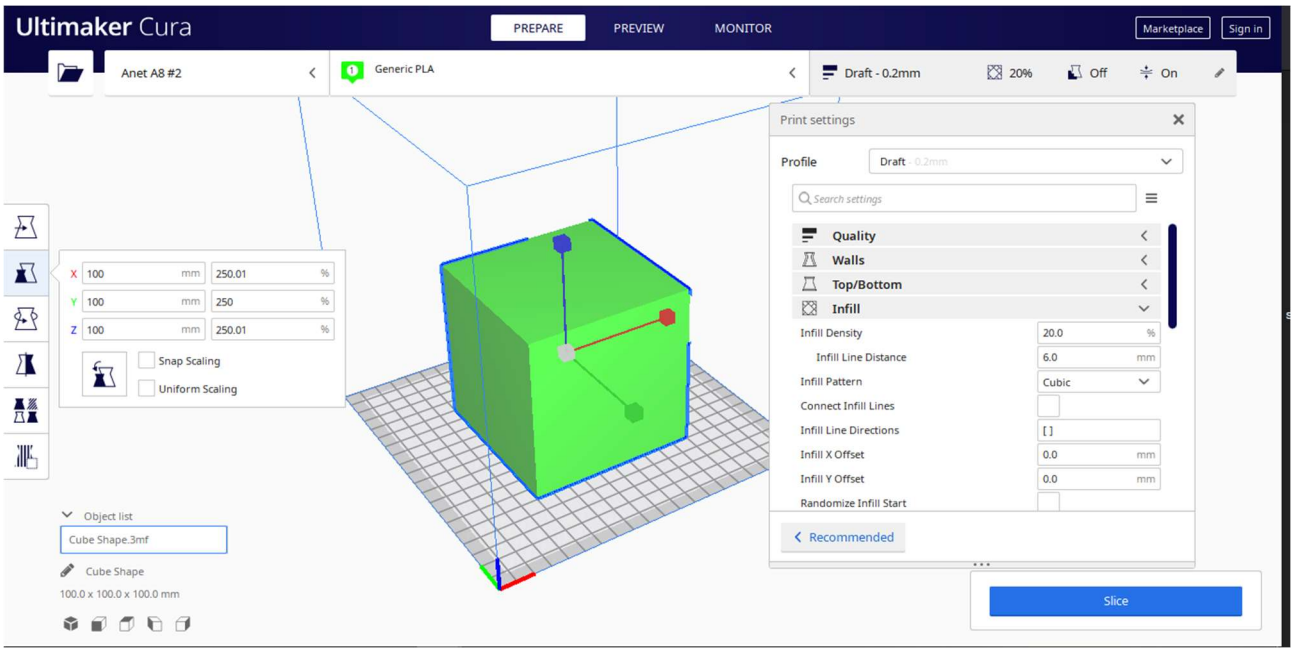


*Imagen 16. Interfaz Ultimaker Cura, visual de un cubo sólido de 10x10 cm*

En esta siguiente imagen se puede observar el resultado del laminado y, en la parte derecha se puede comprobar que el número de capas generadas es 499. Además, una de las grandes mejoras del programa aparece en la parte inferior derecha de la imagen; una aproximación del tiempo necesario para la impresión teniendo en cuenta las especificaciones de la máquina.

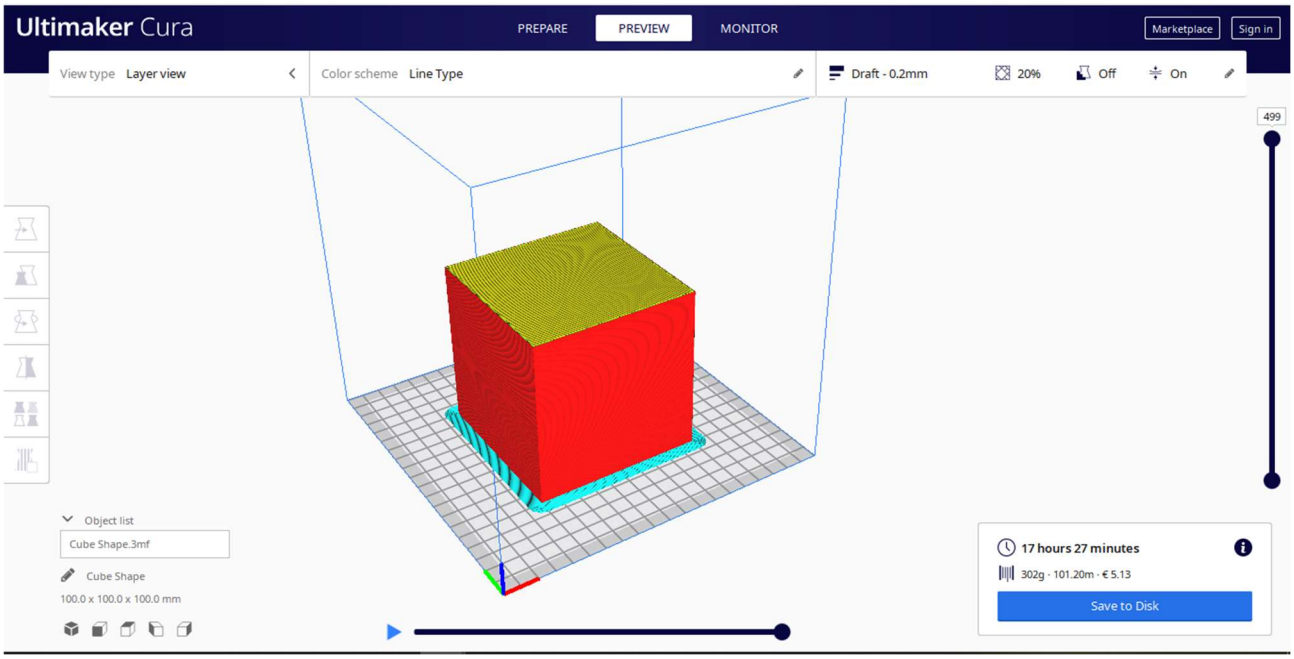


*Imagen 17. Interfaz Ultimaker Cura, previsualización del laminado de un cubo de 10x10 cm*

Esta última imagen muestra el mensaje que se imprime en la consola al terminar el proceso de laminado. En este caso el método, que incluye la generación de las capas, las intersecciones y los polígonos finales, ha tardado 2.15 s aproximadamente.

```
2021-07-03 14:11:11,056 - DEBUG - [MainThread] CuraEngineBackend.CuraEngineBackend._onSlicingFinishedMessage [737]: Slicing took 2.1512255668640137 seconds
2021-07-03 14:11:11,057 - DEBUG - [MainThread] CuraEngineBackend.CuraEngineBackend._onSlicingFinishedMessage [738]: Number of models per buildplate: {0: 1}
2021-07-03 14:11:11,059 - DEBUG - [MainThread] CuraEngineBackend.CuraEngineBackend._onSlicingFinishedMessage [752]: See if there is more to slice...
```

*Imagen 18. Texto de consola, resultado del laminado de un cubo de 10x10 cm utilizando Ultimaker Cura*

# 6. Conclusiones

En este apartado tomaremos la información y los resultados de los apartados anteriores, e intentaremos consensuar su significado práctico. En el apartado anterior, hemos podido comprobar cómo se estructuraban los métodos utilizados por los programas y por qué estrategia optaban cada uno. En el caso de *PrusaSlicer* tienen una estrategia cuya lógica reside en hacer cada parte del proceso general lo más rápido posible para pasar a la siguiente parte. En este caso el mayor problema que hemos detectado es que, al hacer esto; algo o alguien se tiene que encargar de recoger y unir el resultado producido en cada parte, y por supuesto este algo o alguien va a trabajar cada vez más rápido hasta que llegue a su límite. A partir de ese momento la velocidad del método no variará, generando así un cuello de botella.

Por otra parte, la estrategia utilizada por *Ultimaker Cura* tiene una lógica más conservadora, la idea sería dividir a los trabajadores en grandes secciones y darles más trabajo individual. De esta manera cada vez que terminen ya tienen el resultado final y pueden pasar a otra sección y repetir el proceso. Además, en este caso son los propios trabajadores los que organizan los resultados y no hay algo o alguien a cargo de ello sino que todos se encargan de eso. Si de personas se tratara podría descontrolarse y por eso el primer modelo es más utilizado, sin embargo, cuando hablamos de una máquina podemos asumir sin riesgo a equivocarnos que aquellas instrucciones que le hayamos dado se ejecutarán sin percances.

Añadiendo a lo ya comentado, debemos tener en cuenta que en el primer método se emplea mucho el uso de llamadas a funciones y métodos en repetidas ocasiones. Esto supone un aumento considerable del tiempo de computación ya que los datos se deben copiar y devolver los resultados. El tiempo que se invierte en recibir y enviar datos no se invierte en resolver el problema. Esto se palía considerablemente en el segundo método escribiendo todas las instrucciones en un mismo método lo que evita llamadas múltiples a otras funciones.

En resumen, aunque ambos programas hacen uso de la computación paralela para llevar a cabo sus tareas, y reparten la carga entre los diferentes hilos de procesamiento que tienen a su disposición, después del análisis realizado llegamos a la conclusión que

el programa de laminado *Ultimaker Cura*, hace un trabajo más limpio y eficaz que su competidor *PrusaSlicer*.

La computación paralela es una herramienta muy potente pero que aumenta considerablemente la dificultad de la programación. Añade así otra capa de entendimiento que debe tenerse en cuenta a la hora de diseñar y concebir el funcionamiento del código. En los casos expuestos anteriormente es claro que se ha tenido en cuenta, sin embargo, en un caso se ha tenido en cuenta para ser más rápido que un programa que no use varios hilos; y en otro caso se ha tenido en cuenta para que sea lo más rápido posible, y eso puede marcar la diferencia.

# 7. Bibliografía

[1] All3DP. “Los 11 tipos de impresoras.” *All3DP*, 2020. *All3DP*, https://all3dp.com/es/1/tipos-de-impresoras-3d-tecnologia-de-impresion-3d/. Accessed 07 06 2021.

[2] Autodesk. “¿Qué es la impresión 3D?” *AUTODESK*, https://latinoamerica.autodesk.com/solutions/3d-printing. Accessed 07 06 2021.

[3] bitfab. “La Historia de la impresión 3D.” *bitfab*, 2021, https://bitfab.io/es/blog/historia-impresion-3d/. Accessed 07 06 2021.

[4] Greguric, Leo. “Historia de la impresión 3D: fechas clave.” *All3DP*, 2020, https://all3dp.com/es/2/impresion-3d-historia-fechas-clave/. Accessed 14 06 2021.

[5] Nardi, Tom. “3D printing: the past and the future of prusa's slicer.” *Hackaday*, 24 05 2019, https://hackaday.com/2019/05/24/3d-printering-the-past-and-future-of-prusas-slicer/. Accessed 07 06 2021.

[6] Pearson, Aaron. *Historia de la impresión 3D*. 2021. *Startasys*, https://www.stratasys.com/mx/explore/article/3d-printing-history. Accessed 07 06 2021.

[7] Prusa3D. “About-us.” *Prusa3D*, https://www.prusa3d.com/about-us/. Accessed 07 06 2021.

[8] Prusa3D. *TriangleMesh.cpp*. *GitHub*, https://github.com/slic3r/Slic3r/blob/08b32857221e7c706875253ddcbc5a9a94d9f1b9/xs/src/libslic3r/TriangleMesh.cpp#L932. Accessed 17 02 2021.

[9] RepRap. “Skeinforge.” *RepRapWiki*, https://reprap.org/wiki/Skeinforge. Accessed 04 07 2021.

[10] Slic3r. “About.” *Slic3r*, https://slic3r.org/about/. Accessed 02 07 2021.

[11] 3DSOURCED. “The Complete History of 3D Printing: From 1980 to 2021.” *3DSOURCED*, https://www.3dsourced.com/guides/history-of-3d-printing/. Accessed 19 06 2021.

[12] Ultimaker. “Ultimaker turns 10: A look back.” *Talking Additive*, https://www.talkingadditive.com/episodes/episode-25-ultimaker-turns-10-a-look-back. Accessed 07 06 2021.

[13] Ultimaker Cura. *slicer.cpp*. *GitHub*, https://github.com/Ultimaker/CuraEngine/blob/763c299fa8e494ddc5a75a9910208e542127ba48/src/slicer.cpp#L817. Accessed 30 06 2021.

[14] Wikipedia. “Cura (software).” *Wikipedia la enciclopedia libre*, https://en.wikipedia.org/wiki/Cura_(software). Accessed 07 06 2021.

[15] Wikipedia. “Impresión 3D.” *Wikipedia la enciclopedia libre*, https://es.wikipedia.org/wiki/Impresi%C3%B3n_3D. Accessed 07 06 2021.

# 8. Índice de imágenes