

UNIVERSIDADE FEDERAL DO RIO DE JANEIRO
INSTITUTO DE MATEMÁTICA
CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

ARTHUR SILVA DE ALMEIDA
LUIZ BERNARDO LEVENHAGEN ALARCON DA FONSECA
RENAN RAMALHO PEIXOTO COLMAN DA SILVA

BioSlic3r: Uma proposta de melhoria do software Slic3r para suporte à bioimpressão 3D

RIO DE JANEIRO

2019

ARTHUR SILVA DE ALMEIDA

LUIZ BERNARDO LEVENHAGEN ALARCON DA FONSECA

RENAN RAMALHO PEIXOTO COLMAN DA SILVA

BioSlic3r: Uma proposta de melhoria do software Slic3r para suporte à bioimpressão 3D

Trabalho de conclusão de curso de graduação apresentado ao Departamento de Ciência da Computação da Universidade Federal do Rio de Janeiro como parte dos requisitos para obtenção do grau de Bacharel em Ciência da Computação.

Orientadora: Profa. Mônica Ferreira da Silva

Co-orientador: Prof. Eber Assis Schmitz

RIO DE JANEIRO

2019

A447b

Almeida, Arthur Silva de

BioSlic3r: uma proposta de melhoria do software Slic3r para suporte à bioimpressão 3D / Arthur Silva de Almeida, Luiz Bernardo Levenhagen Alarcon da Fonseca, Renan Ramalho Peixoto Colman da Silva. – 2019.

81 f.

Orientadora: Mônica Ferreira da Silva.

Orientador: Eber Assis Schmitz.

Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação) - Universidade Federal do Rio de Janeiro, Instituto de Matemática, Bacharel em Ciência da Computação, 2019.

1. Impressão 3D. 2. Bioimpressão 3D. 3. Engenharia de tecidos. 4. Slic3r. 5. Open source. 6. Software. I. Fonseca, Luiz Bernardo Levenhagen Alarcon da. II. Silva, Renan Ramalho Peixoto Colman da. III. Silva, Mônica Ferreira da (Orient.). IV. Universidade Federal do Rio de Janeiro, Instituto de Matemática. V. Título.

ARTHUR SILVA DE ALMEIDA

LUIZ BERNARDO LEVENHAGEN ALARCON DA FONSECA

RENAN RAMALHO PEIXOTO COLMAN DA SILVA

BioSlic3r: Uma proposta de melhoria do software Slic3r para suporte à Bioimpressão 3D

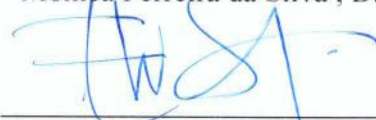
Trabalho de conclusão de curso de graduação apresentado ao Departamento de Ciência da Computação da Universidade Federal do Rio de Janeiro como parte dos requisitos para obtenção do grau de Bacharel em Ciência da Computação.

Aprovado em 18 de dezembro de 2019.

BANCA EXAMINADORA:



Mônica Ferreira da Silva, DsC (UFRJ)



Eber Assis Schmitz, PhD (UFRJ)



Renato Montaleão Brum Alves, MSc (Petrobras)



Élton Carneiro Marinho, MSc (UFRJ)

DEDICATÓRIA

Dedico esse trabalho à memória de meu pai, Robson, que sempre incentivou meu interesse em tecnologia e informática. Agradeço à minha família, meus tios, primos e minha avó por todo apoio. A todos os amigos que de alguma maneira me ajudaram durante esses muitos anos. Gostaria de agradecer à minha orientadora, ao orientador, professores da banca e aos amigos que trabalharam comigo nesse projeto. Por fim, mais do que agradecer, eu gostaria de dividir esse projeto com minha mãe, Carla, que esteve ao meu lado sempre, apoiando e aconselhando em todas as minhas decisões.

- ARTHUR SILVA DE ALMEIDA

Gostaria de dedicar primeiramente à Deus. Também à minha família, por todo suporte em todos os momentos. Aos meus incansáveis amigos Renan e Arthur. Por fim, porém não menos importante, à todos que de alguma forma me incentivaram e me ajudaram a chegar até aqui.

- LUIZ BERNARDO LEVENHAGEN
ALARCON DA FONSECA

Agradeço à todos que estiveram presentes no meu processo acadêmico, à minha família que desde o início valorizou e possibilitou meu aprendizado investindo em mim e no meu potencial, principalmente meus pais, minha avó e meu irmão que me apoiaram desde o início da minha empreitada. Foram anos para chegar até aqui, algumas dificuldades mas sempre tive o estímulo necessário para continuar. Gratidão à todos os meus amigos que compartilharam comigo todas as conquistas que adquiri. Além disso, gostaria de agradecer à orientadora Mônica que dispôs de sua experiência para aprimorar e auxiliar no fim desta etapa, também ao grupo deste projeto que não desistiu de tentar, por fim, ao coorientador e aos professores presentes na banca.

- RENAN RAMALHO PEIXOTO
COLMAN DA SILVA

Agradecimentos

Agradecemos aos nossos familiares que nos apoiaram até aqui e que foram a nossa fonte de inspiração. Somos gratos aos colegas de Universidade que lutaram junto conosco todos os dias. Aos amigos que não deixaram o cansaço nos vencer. Aos nossos orientadores Monica Ferreira e Eber Assis Schmitz que foram incansáveis em suas orientações, pesquisas e revisões. Nosso muito obrigado à Universidade Federal do Rio de Janeiro por nos proporcionar o melhor ambiente educacional. Por último mas não menos importante agradecemos à Janaina Dernowsek, Ph.D. que nos auxiliou durante o processo de desenvolvimento do *software* estudado.

Epígrafe: Nós só podemos ver um pouco do futuro, mas o suficiente para perceber que há muito a fazer.

ALAN MATHISON TURING

RESUMO

Manufatura aditiva, também conhecida como impressão 3D, consiste na criação de objetos físicos resultantes da adição de camadas sucessivas de material. Essa tecnologia está evoluindo e crescendo rapidamente. Atualmente, a impressão 3D é amplamente utilizada no mundo, em diversas áreas de atuação tais como customização de peças em larga escala, criação de protótipos de projetos em diversas indústrias, aplicações em setores como setor alimentício, automotivo, aéreo, vestuário e até mesmo na saúde.

Na área da saúde, utiliza-se comumente a nomenclatura bioimpressão 3D que pode ser definida pela fabricação aditiva de biomateriais utilizando células e outros produtos biológicos. Essa área de atuação veio para complementar e propor melhorias ao estudo da engenharia de tecidos, é uma abordagem inovadora e cada vez mais utilizada na biomedicina com potencial para projetar tecidos, além de criar protótipos funcionais para triagem de medicamentos, pesquisa toxicológica e também transplante de órgãos

Esta monografia tem dois grandes objetivos, o primeiro, apresentar uma Revisão *quasi* Sistemática da Literatura e um estudo da bioimpressão 3D, analisando artigos acadêmicos aplicando processos de avaliação que determinam o estado atual da bioimpressão e suas lacunas, organizadas de forma coerente para responder à problemática inicial da pesquisa.

Nosso segundo propósito é apresentar uma melhoria ao *software* de fatiamento Slic3r, muito utilizado na área de biofabricação 3D para fatiamento de modelos e configuração de parâmetros de bioimpressão porém, que atualmente apresenta pouco suporte à essa tecnologia por ter como objetivo principal a impressão 3D convencional. Este objetivo é inspirado no ideal da comunidade *open source* que é contribuir com a disseminação de conhecimento de forma ampla e gratuita.

O *software* Slic3r, por ser um *software open source*, permite que pessoas com determinado conhecimento contribuam e desenvolvam novas funcionalidades ao programa, pensando nisso e em como notamos uma quantidade muito pequena de *softwares* voltados à bioimpressão 3D, realizamos uma análise e levantamento da atual situação da área de biofabricação e o desenvolvimento da nova funcionalidade no Slic3r adicionando suporte à bioimpressão 3D.

Palavras-chave: impressão 3d. bioimpressão 3D. engenharia de tecidos. slic3r. open source. software.

ABSTRACT

Additive manufacturing, also known as 3D printing, is the creation of physical objects resulting from the addition of successive layers of material. This technology is evolving and growing fast. Today, 3D printing is widely used around the world in a variety of areas such as customizing large-scale parts, designing prototypes in a variety of industries, applications in industries such as the food industry, automotive, aerospace engineering, clothing and even in the health industry.

In the health industry, the 3D bioprint nomenclature is commonly used and can be defined by the additive manufacturing of biomaterials using cells and other biological products. This area of work came to complement and propose improvements to the study of tissue engineering. It is an innovative approach and is increasingly used in biomedicine with potential for tissue design, as well as creating functional prototypes for drug screening, toxicological research and also organ transplant.

This research has two major objectives, the first it to present a *quasi*-systematic literature review and a study of 3D bioprinting, analyzing academic papers by applying evaluation processes that determine the current state of bioprinting and its gaps, coherently organized to respond to the initial research problem.

Our second purpose is to present an improvement to the Slic3r slicing software, widely used in the area of 3D biofabrication for model slicing and setting bioprinting parameters, but currently has little or no support for this technology because its main purpose is conventional 3D printing. This goal is inspired by the open source community ideal, which is to contribute to the dissemination of knowledge widely and for free.

The Slic3r software, as an open source software, allows people with certain knowledge to contribute and develop new features to the program, thinking about this and how we've noticed a very small amount of 3D bioprinting software papers, we performed an analysis and surveyed the current situation of the biofabrication and developed a new functionality in Slic3r by adding support for 3D bioprinting.

Keywords: 3d printing. 3d bioprinting. tissue engineering. slic3r. open source. software.

LISTA DE ILUSTRAÇÕES

Quadro 1 - Critérios de inclusão e exclusão.....	20
Figura 1 - Linha do tempo da Revisão Sistemática da Literatura.....	21
Gráfico 1 - Resultados por sites de busca	21
Figura 2 - Tríplice da engenharia de tecidos (O'Brien, 2011)	24
Figura 3 - Ilustração da técnica FDM	26
Figura 4 - Objeto impresso utilizando a técnica FDM	26
Figura 5 - Ilustração da técnica de estereolitografia (SLA)	27
Figura 6 - Objeto no processo de impressão utilizando a técnica SLA	27
Figura 7 - Ilustração da técnica SLS	28
Figura 8 - Peça sendo retirada do recipiente após ser impressa utilizando a técnica SLS	28
Figura 9 - Rolos de filamento PLA	29
Figura 10 - Fluxo das etapas importantes de impressão 3D.....	32
Figura 11 - Diferentes técnicas de bioimpressão 3D	33
Figura 12 - Uma representação da bioimpressão a jato de Biotinta, onde a partir de um reservatório, gotas de biotinta são formadas pela ação de um atuador piezoelétrico(b) ou térmico(a) e são depositadas em uma placa de coleta.....	34
Figura 13 - Formas de microextrusão de material	35
Figura 14 - Diagrama esquemático da bioimpressão baseada em sinterização à laser	36
Figura 15 - Formação de esferóides pelo método Hanging Drop	38
Figura 16 - Processo de bioimpressão 3D.....	39
Figura 17 - Biorreator de perfusão virtual.....	40
Figura 18 - BioEdPrinter - Bioimpressora básica da startup BioEdTech	40
Figura 19 - Fluxo das etapas importantes de bioimpressão 3D	41
Figura 20 - Tela do software PetriPrinter.....	44
Figura 21 - Guia para placas de Petri desenvolvido com o software PetriPrinter	44
Figura 22 - Fluxo atual de bioimpressão	45
Figura 23 - Fluxo alcançado com o desenvolvimento inicial do BioSlic3r	45

Figura 24 - Fluxo esperado para trabalhos futuros com etapas integradas no fatiador Slic3r..	45
Figura 25 - Explicação dos diretórios do projeto Slic3r pela documentação oficial	48
Figura 26 - Classe MainViewModel	49
Figura 27 - Classe IOTools	49
Figura 28 - Fluxo antigo de bioimpressão utilizando o Slic3r, gCodeEditor e PetriPrinter....	53
Figura 29 - Novo fluxo de bioimpressão após melhoria da funcionalidade para o Slic3r.....	54

LISTA DE TABELAS

Tabela 1 - Construção da <i>string</i> de busca	19
Tabela 2 - Bases com suas correspondentes <i>strings</i> de busca	20
Tabela 3 - Tabela comparativa das modalidades de bioimpressão 3D	37
Tabela 4 - Ambiente de Teste e Desenvolvimento	46

LISTA DE SIGLAS

3D	Três dimensões
SMEM	Sistema Micro-Eletro-Mecânico
JTC	Jato de Tinta Contínuo
DSD	Deposição Sob Demanda
MEC	Matriz Extracelular
SLS	Sinterização Seletiva à Laser
SLA	Estereolitografia
FDM	Modelagem por deposição fundida
TI	Tecnologia da Informação
PTF	PetriTask File

SUMÁRIO

1	INTRODUÇÃO	14
1.1	MOTIVAÇÃO	14
1.2	OBJETIVOS GERAIS	15
1.3	OBJETIVOS ESPECÍFICOS	15
1.4	ORGANIZAÇÃO DA MONOGRAFIA	15
1.5	TRABALHOS RELACIONADOS.....	16
1.5.1	O papel da tecnologia da informação no futuro da biofabricação 3D	16
1.5.2	Novas ferramentas de software para bioimpressão baseada em hidrogel	16
2	METODOLOGIA DA PESQUISA	17
2.1	INTRODUÇÃO	17
2.2	PROTOCOLO QUASI-SISTEMÁTICO	17
2.3	PESQUISA PARA DESENVOLVIMENTO DO NOVO ALGORITMO	21
3	REFERENCIAL TEÓRICO	22
3.1	ENGENHARIA DE TECIDOS	22
3.1.1	Motivação dos estudos na área	22
3.1.2	Definição da tecnologia	22
3.2	IMPRESSÃO 3D	24
3.2.1	Técnicas de impressão 3D	24
3.2.2	Tipos de materiais na manufatura aditiva	28
3.2.3	Aplicações.....	29
3.2.4	As 4 grandes etapas da impressão 3D	30
3.3	BIOIMPRESSÃO 3D	32
3.3.1	Técnicas de bioimpressão	33
3.3.2	As 4 grandes etapas da bioimpressão 3D	37
3.3.3	Softwares de fatiamento para bioimpressão 3D	41
3.3.4	Desafios da bioimpressão	42
4	PROPOSTA DE MELHORIA NO SOFTWARE DE FATIAMENTO	42
4.1	BIOSLIC3R	42
4.2	FLUXOS DE CONTRIBUIÇÃO DO BIOSLIC3R	44
5	DESENVOLVIMENTO DO ALGORITMO	45
5.1	AMBIENTE DE DESENVOLVIMENTO	45
5.2	DESCRIÇÃO DO DESENVOLVIMENTO.....	48
5.3	TESTES, RESULTADOS E AVALIAÇÃO	49
6	CONCLUSÃO	51
6.1	PRINCIPAIS CONTRIBUIÇÕES	51

6.2	VISÃO GERAL	52
6.3	PRINCIPAIS DIFICULDADES ENCONTRADAS	52
6.4	TRABALHOS FUTUROS	52
	REFERÊNCIAS	54
	APÊNDICE A – FUNÇÕES PRINCIPAIS	57

1 INTRODUÇÃO

1.1 MOTIVAÇÃO

A Impressão 3D é uma forma de tecnologia de manufatura aditiva onde um modelo tridimensional é criado por sucessivas camadas de material. Estudos sobre estas tecnologias já têm sido feitos desde 1981 (SHAHRUBUDIN, 2019), mas eles só começaram a se tornar mais populares a partir do ano de 2009, pois, com a queda da patente da tecnologia FDM diversas empresas começaram a desenvolver suas próprias impressoras, consequentemente barateando os custos no mercado.

De maneira similar, a engenharia de tecidos ou biofabricação, vem sendo um tópico relevante nas últimas décadas (FERMEIRO, 2015) em virtude dos problemas enfrentados pela sociedade no que tange a medicina regenerativa, a escassez de órgãos para transplante e os grandes tempos de espera em filas de transplantes de órgãos que acontecem principalmente pela recusa familiar à doação (ABTO, 2013).

Este campo da medicina, em particular, utiliza conhecimentos multidisciplinares, envolvendo a medicina, engenharia, química e física para desenvolver tecidos artificiais a fim de melhorar tecidos danificados ou até mesmo fabricar por completo um órgão ou tecido. Sendo assim, devido à crescente demanda pela melhoria das técnicas de biofabricação, notou-se que os conceitos de impressão 3D poderiam ser aplicados nesta área, com isso surgiu a bioimpressão 3D.

A bioimpressão 3D pode parecer um conceito novo e extremamente complexo para a maioria das pessoas, mas com a popularização dos *hardwares* de impressão 3D e a comunidade *open source* contribuindo com *softwares*, estão disponíveis *online* diversas fontes de conhecimento sobre essa temática, bem como tutoriais ensinando a construir uma bioimpressora de baixo custo e como contribuir com a comunidade da melhor maneira possível.

Algo que ainda é escasso e carece de muitas alternativas é a quantidade de *softwares* livres com suporte à bioimpressão (DERNOWSEK, 2017).

Uma alternativa utilizada pela maioria das pessoas que se interessam por bioimpressão 3D é recorrer aos *softwares* de impressão 3D convencional e, por tentativa e erro, criar configurações compatíveis com suas bioimpressoras.

Isto é uma tarefa muito custosa, não somente de tempo, mas também de material, consequentemente gerando custos monetários, pois são necessários diversos testes minuciosos para calibração, configuração de parâmetros, testes de impressão na prática, entre muitos outros.

Analisando esse cenário, notamos uma grande necessidade de melhoria nestes *softwares* de impressão 3D, de maneira que passem a ter suporte para bioimpressão 3D.

1.2 OBJETIVOS GERAIS

Devido a necessidade de maior suporte na engenharia de tecidos, visa-se com este trabalho analisar o campo de tecnologias de impressão e bioimpressão 3D a fim de propor uma possível contribuição para o Slic3r, um *software open source* utilizado para fatiamento de modelos 3D e geração de arquivos GCODE interpretados pelas impressoras 3D, de modo que seja possível utilizá-lo para bioimpressão 3D de maneira mais simples, facilitando a configuração dos parâmetros necessários para a bioimpressão e reduzindo a necessidade do uso de mais de um *software* ao mesmo tempo.

1.3 OBJETIVOS ESPECÍFICOS

- Apresentar os principais conceitos e relacionamento entre às áreas de engenharia de tecidos, impressão 3D e bioimpressão 3D
- Apresentar uma revisão *quasi* sistemática de literatura analisando a temática de bioimpressão 3D
- Apresentar uma contribuição ao projeto *open source* do *software* Slic3r, implementando no programa uma nova função voltada à bioimpressão 3D baseada na integração dos *softwares* PetriPrinter, Gcode Editor e Slic3r.

1.4 ORGANIZAÇÃO DA MONOGRAFIA

Esta monografia está organizada da seguinte maneira:

O capítulo 2 descreve a metodologia da pesquisa, as formas como analisamos e chegamos ao nosso objetivo de pesquisa.

O capítulo 3 descreve o referencial teórico e as descrições e complementações existentes entre as tecnologias de engenharia de tecidos, impressão 3D e bioimpressão 3D.

O capítulo 4 descreve a proposta do algoritmo a ser desenvolvido.

O capítulo 5 descreve a implementação do algoritmo desenvolvido.

O capítulo 6 descreve a conclusão chegada após o desenvolvimento da pesquisa e considerações finais.

1.5 TRABALHOS RELACIONADOS

O papel que a tecnologia da informação tem nas grandes áreas de pesquisa é algo que torna-se cada vez maior. A necessidade do uso de TI na área de biofabricação não é algo completamente novo e existem artigos que exaltam essa necessidade, os de maior relevância para este estudo serão citados e brevemente explicados nos subitens abaixo, são eles:

1.5.1 O papel da tecnologia da informação no futuro da biofabricação 3D

O artigo (DERNOWSEK, 2016) publicado no “Journal of 3D Printing in Medicine” analisa o papel da TI no futuro da biofabricação 3D, analisando as demandas da bioimpressão 3D à tecnologia da informação, os avanços atuais e as necessidades futuras da área.

Nele podemos notar como a TI influenciou positivamente no avanço da biofabricação nas últimas décadas, seja no aprimoramento das técnicas de engenharia de tecidos, modelagens de objetos 3D ou simulação computacional, além de nos mostrar as perspectivas futuras esperadas na correlação das duas áreas.

1.5.2 Novas ferramentas de software para bioimpressão baseada em hidrogel

Segundo o artigo (ROBU, 2019), o resultado da bioimpressão, depende não só das células já posicionadas mas também do biomaterial que ainda será depositado. O trabalho em questão apresenta novas informações e ferramentas para prever a formação da estrutura pós-impressão em construções bioimpressas. Foi feita uma adaptação em um algoritmo de simulação que leva em consideração as energias da interação entre todos os elementos do sistema.

Também foi criado um módulo que gera automaticamente os modelos 3D de construções de tecidos biofabricados. Portanto, abriu-se a possibilidade de integrar diversas

arquiteturas de construções de tecidos bioimpressos. Para validar os novos componentes de *software*, foram gerados dois modelos usados em experimentos, foi simulado sua evolução encontrando um equilíbrio qualitativo entre experimentos e simulações.

2 METODOLOGIA DA PESQUISA

2.1 INTRODUÇÃO

Neste capítulo, abordamos a metodologia que guiou a execução desta pesquisa, salientando seu caráter multimétodo, ou seja, para atingir o objetivo almejado, foi necessário que esta pesquisa fosse dividida em duas etapas, onde inicialmente foi feita uma Revisão *quasi* Sistemática da Literatura (RqSL) e com esta primeira fase foi notado uma grande necessidade de um *software* que fosse focado em Bioimpressão 3D e após constatarmos esta necessidade começamos o processo de desenvolvimento de uma melhoria para um *software* com o objetivo de aprimorar o processo de pré-bioimpressão 3D, os fatiadores.

2.2 PROTOCOLO *QUASI-SISTEMÁTICO*

Revisão *quasi* Sistemática da Literatura

Foi realizada uma Revisão *quasi* Sistemática da Literatura (RqSL), em um primeiro momento com o objetivo de compreender como o processo de bioimpressão 3D poderia ser melhorado.

O protocolo utilizado para Revisão *quasi* Sistemática da Literatura (RSL) foi:

1. Perguntas da RSL
2. Base de Dados
3. *Strings* de Busca
4. Critérios de Inclusão/Exclusão
5. Resultados

1 - Pergunta da RqSL

Como o *software* Slic3r pode ser modificado a fim de **melhorar sua agregação de valor** à área de bioimpressão 3D e engenharia de tecidos?

2 - Bases de dados

- ScienceDirect
- IEEE

- Scopus
- SemanticScholar
- Springer

3 - *String* de busca

Para a construção da *string* de busca final da RqSL foram necessárias algumas pré-análises explicitadas na Tabela 1.

Tabela 1 - Construção da *string* de busca

Tentativa	<i>String</i>	Análise
1	(Title: "bioprint" OR "bioprinting" OR "bioprinter" OR "3d print") AND (abstract: "stem cell" OR "tissue")	Baixa precisão resultados não satisfatórios
2	("bioprint" OR "bioprinting" OR "bioprinter") AND ("stem cell" OR "tissue")	Muito focado na parte de biologia e medicina.
3	(bioprint OR "bio-print") AND (software OR computation) AND (predict OR simulation)	Constatação de que a grande maioria dos artigos analisados observaram a necessidade de um <i>software</i> focado em Bioimpressão 3D

Fonte: Desenvolvida pelos autores

Foi utilizada a *string* de busca: ("bioprint" OR "bio-printing") AND (computation OR software) AND ("simulation" OR "prediction"), sofrendo alterações para cada base de dados para poder corresponder à mesma busca desejada, como pode ser notado na Tabela 2..

As bases Science Direct e Semantic Scholar foram incluídas por serem fontes de pesquisa multidisciplinares e com isso o tema seria melhor abordado, já a IEEE foi destacada justamente pelo seu foco na área de tecnologia, Scopus e Springer Link foram selecionadas pois são uma das maiores bases de dados.

Tabela 2 - Bases com suas correspondentes *strings* de busca

Banco de dados	String de busca
Science Direct	(bioprint OR "bio-print") AND (software OR computation) AND (predict OR simulation)
IEEE	(bioprint OR "bio-print") AND (software OR computation) AND (predict OR simulation)
Scopus	ALL ((bioprint OR "bio-print") AND (software OR computation) AND (predict OR simulation))
SpringerLink	(bioprint OR "bio-print") AND (software OR computation) AND (predict OR simulation)
SemanticScholar*	(bioprint OR "bio-print") AND (software OR computation) AND (predict OR simulation)

Fonte: Desenvolvida pelos autores

**SemanticScholar se trata de um site que usa inteligência artificial para buscar em dezenas de bases científicas, dentre elas PubMed, ACM, ACL, SpringerNature, entre outras.*

4 - Critérios de Inclusão/Exclusão

A fim de calibrar e tornar a busca mais precisa e acurada utilizamos os critérios de inclusão e exclusão que podem ser visualizados no Quadro 1.

Quadro 1 - Critérios de inclusão e exclusão

Inclusão	Exclusão
<ol style="list-style-type: none"> 1. Trabalhos publicados de 2014 em diante; 2. Busca por artigos e conferências; 3. Trabalhos em inglês. 	<ol style="list-style-type: none"> 1. Trabalhos publicados antes de 2014; 2. Trabalhos com acesso privado; 3. Books, Protocolos, Capítulos; 4. Trabalhos em outras línguas que não o Inglês; 5. Trabalhos duplicados.

Fonte: Desenvolvida pelos autores

5 - Resultado

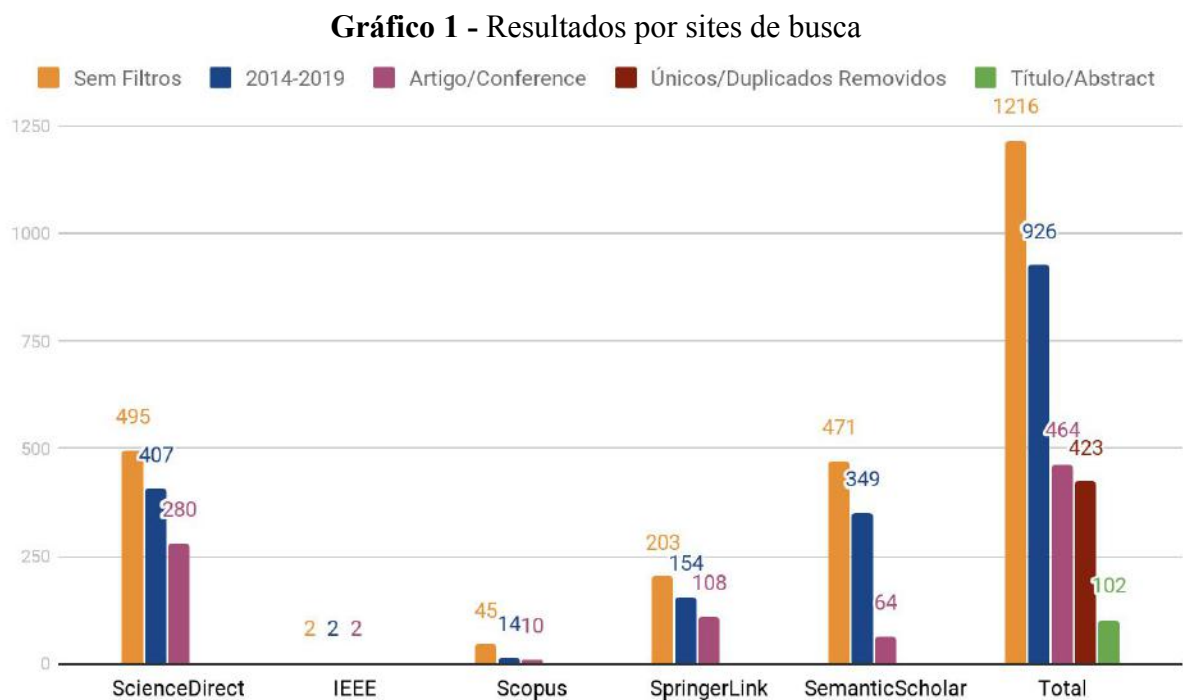
Após aplicar os critérios de inclusão e exclusão explicitados acima, obtivemos os resultados apresentados a seguir na Figura 1:



Figura 1 - Linha do tempo da Revisão Sistemática da Literatura.

Fonte: Desenvolvido pelos autores, 2019.

Para melhor visualização, o Gráfico 1 foi elaborado com o passo-a-passo dos resultados divididos por *site* de busca.



Fonte: Desenvolvimento dos autores, 2019.

Sabendo que o processo de revisão sistemática pode ser comparado à montagem de um quebra-cabeça (PETTICREW; ROBERTS, 2006), no qual os artigos representam as peças e os processos de avaliação determinam se essas peças fazem ou não parte da figura final que se quer montar, ainda dependemos que essas peças consigam ser organizadas de forma coerente para responder à problemática inicial da pesquisa.

Em vista dos argumentos apresentados e após avaliação dos títulos, *abstracts* e trabalhos futuros, ficou constatado que ainda não foi desenvolvida nenhuma solução de *software open source* específico para bioimpressão 3D, assim sendo, começamos a desenvolver uma melhoria do *software* Slic3r para dar suporte à bioimpressão 3D.

2.3 PESQUISA PARA DESENVOLVIMENTO DO NOVO ALGORITMO

Com o decorrer da pesquisa, foram levantadas ideias para aprimoramento do *software* fatiador Slic3r, dentre as quais são relevantes citar:

- Criação de uma aba na tela principal de forma semelhante à existente, voltada inteiramente para bioimpressão, onde seria possível configurar os parâmetros não somente para impressão baseada em extrusão de material, mas também para outros tipos de bioimpressoras.
- Integração do fatiador diretamente com *softwares* de modelagem BioCAD, para que a modelagem pudesse ser modificada internamente pelo fatiador, a fim de consertar possíveis problemas com o modelo 3D e evitar problemas na impressão.
- Criação de uma aba contendo informações relevantes sobre os parâmetros de bioimpressão customizáveis pelo fatiador.
- Integração do Slic3r com outros softwares facilitadores das etapas de bioimpressão 3D

Após analisarmos as ideias levantadas, comparando complexidade, material disponível para consulta, tempo de desenvolvimento e relevância da contribuição, chegamos à conclusão de que as ideias mais viáveis são as duas últimas citadas anteriormente.

Sendo assim, passamos a buscar outros softwares relacionados com bioimpressão 3D e em conversa com a pesquisadora PhD Janaína Dernowsek, nos foi indicado o uso do *software* PetriPrinter, um *software* que facilita os testes e distribuição de placas de Petri para testes de bioimpressão 3D controlada.

3 REFERENCIAL TEÓRICO

3.1 ENGENHARIA DE TECIDOS

A engenharia de tecidos surgiu da necessidade da criação de tecidos artificiais para reparação de danos e criação de órgãos artificiais para transplante, ela tem como objetivo auxiliar na regeneração e restauração dos tecidos deformados ou lesados, gerando substitutos biológicos capazes de reparar, manter ou aperfeiçoar o desempenho e atividade dos mesmos. (MATAI, 2019).

3.1.1 Motivação dos estudos na área

A engenharia de tecidos combina os princípios de transplante de materiais e células para o desenvolvimento de tecidos ou regeneração endógena. Essa abordagem foi inicialmente seguida para tentar amenizar o espaço enorme entre pacientes portadores de doenças graves com risco de morte e necessitados de transplante de órgãos e o número de órgãos disponíveis para transplante (FURTH, 2014).

Por outro lado, não se resumindo apenas em doenças causadoras de morte, a engenharia de tecidos também atinge outras patologias. Como por exemplo, perdas dentárias devido aos processos patológicos, ainda são uma realidade na clínica odontológica diária, induzindo complicações funcionais e psicológicas ao paciente. Esta complicação também é abordada como possivelmente solucionável pela engenharia tecidual. (MORO, 2018)

3.1.2 Definição da tecnologia

O termo “engenharia de tecidos” foi citado originalmente em 1988, em um *workshop* na Fundação de Ciência Nacional, para indicar “a aplicação de princípios e métodos de engenharia e ciências humanas em virtude do entendimento fundamental do relacionamento estrutura-função em tecidos mamários normais e patológicos e o desenvolvimento de substitutos biológicos para restaurar, manter e melhorar a função tecidual”. Porém, enquanto a área de engenharia de tecidos seja relativamente nova, a ideia de substituir um tecido humano por outro **vai até muito atrás, no século 1.**

O campo de engenharia de tecidos é altamente multidisciplinar e possui profissionais de medicina clínica, engenharia mecânica, ciência dos materiais, genética e disciplinas de engenharia e ciências biológicas.

Essa área se baseia extensivamente no uso de suportes porosos para fornecer o ambiente apropriado para a regeneração de tecidos e órgãos. Esses suportes agem essencialmente como um modelo para a formação de tecidos e são tipicamente irrigados com células e ocasionalmente fatores de crescimento ou sujeitos a estímulos biofísicos na forma de um biorreator; um dispositivo ou sistema onde diferentes tipos de estímulos mecânicos e químicos são realizados nas células.

Esses suportes irrigados por células são, ou colocados em cultura *in vitro* para sintetizar tecidos que serão futuramente implantados no local danificado, ou são implantados diretamente no local danificado, utilizando o próprio sistema do corpo, onde a regeneração do tecido ou órgão é induzido *in vivo*.

Essa combinação de células, sinais e suportes é geralmente referenciada como o tripé da engenharia de tecidos, conforme visto na Figura 2. (O'Brien, 2011)

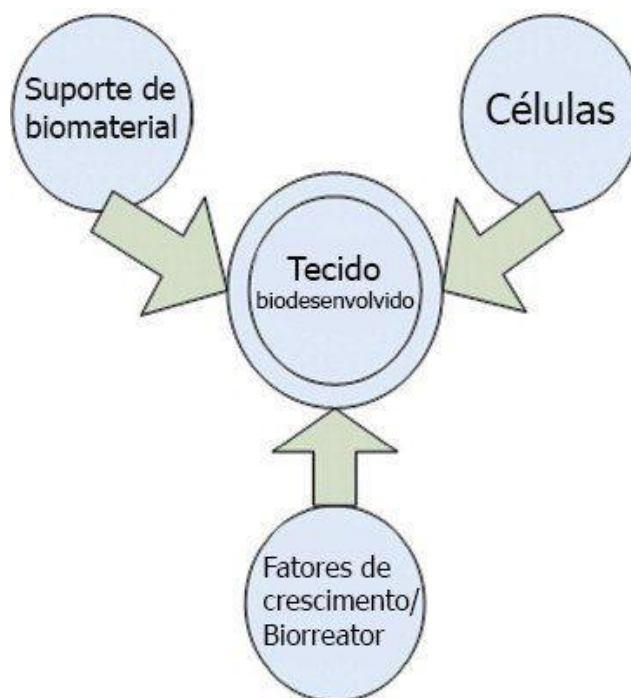


Figura 2 - Trílice da engenharia de tecidos (O'Brien, 2011)

Fonte: (MATERIALSTODAY, 2011). Traduzido pelos autores)

3.2 IMPRESSÃO 3D

A impressão 3D ou manufatura aditiva, pode ser definida como um processo de criação de um objeto a partir da deposição ou sinterização de sucessivas camadas de material. Das diversas vantagens deste processo em comparação com processos de fabricação tradicionais, destacam-se: (i) eficiência: produção rápida e econômica, com baixas quantidades de material residual; (ii) criatividade: ideal para confecção de geometrias complexas; (iii) acessibilidade: preço razoável de máquinas e materiais (SANTANA, 2018).

Desde sua introdução, por Charles Hull, em 1986, quando registrou a patente da técnica de manufatura aditiva utilizando **estereolitografia** (HULL, 1985), o processo de impressão 3D se desenvolveu e foi aprimorado a ponto de termos diversas técnicas diferentes para produção de objetos tridimensionais utilizando os mais diversos materiais, desde materiais plásticos (COELHO, 2018), usados pela grande maioria das pessoas, até mesmo ligas metálicas usadas em processos industriais (DUDA, 2016).

A popularização e aprimoramento da tecnologia se deu, principalmente após a expiração das patentes relacionadas ao processo de Modelagem por Deposição Fundida (Fused Deposition Modeling - FDM), pois uma grande comunidade se desenvolveu voltada à elaboração de sistemas de código aberto, tanto de *software* quanto de *hardware*, e a partir disso, em pouco tempo o preço das máquinas foi reduzido consideravelmente (LIFTON, 2014).

A impressão 3D é de fato uma tecnologia revolucionária e inovadora, com o potencial de mudar a forma como as coisas são produzidas na indústria. No futuro, estima-se que a tecnologia estará cada vez mais rápida e com custos menores, de modo que cada pessoa poderá ter uma impressora em casa e poderá realizar a fabricação de objetos do dia a dia de forma quase instantânea.

3.2.1 Técnicas de impressão 3D

Dentre as diversas técnicas existentes da manufatura aditiva tais como deposição de energia direcionada, fusão de material em pó, fotopolimerização, a que mais se destacou foi a técnica de extrusão de material (SHAHRUBUDIN, 2019).

Essa técnica se popularizou pois é o tipo de impressão realizada por impressoras de preços mais populares, além de utilizar uma matéria prima mais barata e ser uma técnica com baixa curva de aprendizado. Muitas vezes, com máquinas mais sofisticadas, é possível imprimir uma peça com o apertar de poucos botões, de maneira intuitiva.

Impressão baseada em deposição de material fundido

A técnica conhecida como FDM é uma tecnologia baseada em extrusão, a matéria prima utilizada é um filamento, que pode ser de diversos materiais, como diferentes tipos de materiais termoplásticos, materiais mistos com metais, madeira, entre outros. Na utilização, o filamento é aquecido a temperaturas que variam de 150°C a mais de 200°C e depositado por um bico extrusor em uma plataforma plana. As camadas do material depositadas esfriam rapidamente e se solidificam, unindo-se às camadas previamente depositadas. Após isso, a plataforma descende uma distância determinada e mais material é depositado na camada anterior, como pode ser visto nas Figuras 3 e 4. O processo se repete até a formação do objeto final (GIBSON, 2014).

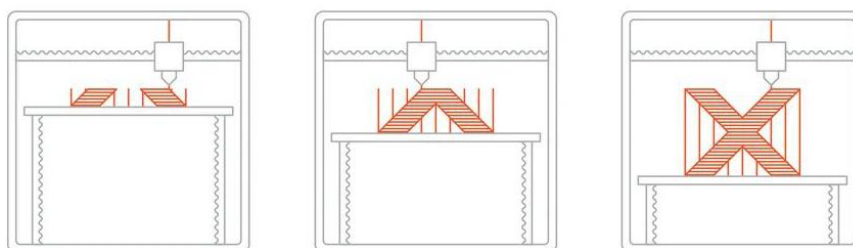


Figura 3 - Ilustração da técnica FDM

Fonte: (CORE77, 2017)

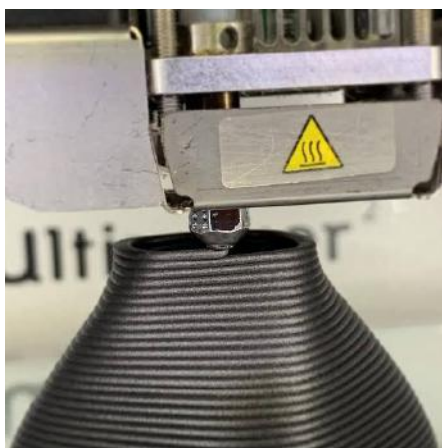


Figura 4 - Objeto impresso utilizando a técnica FDM

Fonte: (3DHUBS, 2016)

Impressão baseada em Estereolitografia

A estereolitografia, técnica pioneira da manufatura aditiva, consiste na utilização de um feixe de laser para enrijecer camadas de resina líquida, a fim de formar um objeto. Ao contrário da técnica FDM, a **SLA** não deposita material em uma plataforma, mas sintetiza as camadas da resina a partir de um recipiente. Quando termina uma camada, a plataforma ascende e outra camada é sintetizada por cima da anterior, se fundindo com a camada previamente sintetizada. Processo ilustrado nas Figuras 5 e 6. Assim como a FDM, o processo se repete até a formação do objeto final.

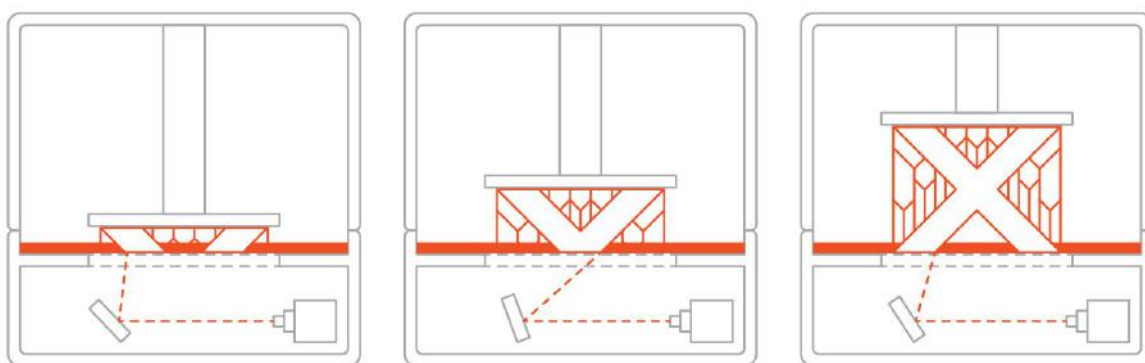


Figura 5 - Ilustração da técnica de estereolitografia (SLA)

Fonte:(CORE77, 2017)

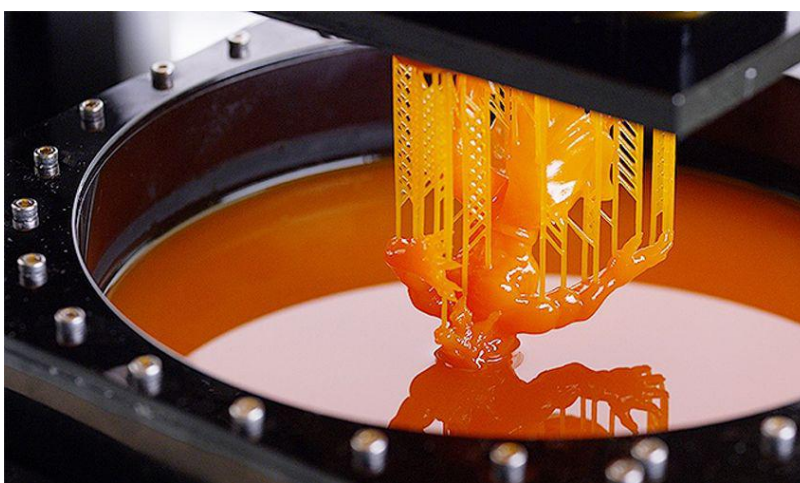


Figura 6 - Objeto no processo de impressão utilizando a técnica SLA

Fonte: (3DNATIVES, 2017)

Impressão baseada em Sinterização Seletiva à Laser

A técnica de Sinterização Seletiva à Laser ou SLS, diferente das técnicas citadas anteriormente, utiliza um recipiente com material em pó como matéria prima. Assim como a técnica SLA, utiliza de um feixe de laser para enrijecer e unir as partículas do material de maneira constante a fim de gerar um objeto 3D. Durante o processo, o laser é apontado para o recipiente e “desenha” um padrão pré definido na superfície, em cima da camada anterior. Isso é iniciado no topo e finalizado na base, de maneira que o objeto é criado na parte interior do recipiente com a matéria prima em pó. Ilustrado nas Figuras 7 e 8. Esta técnica é bastante utilizada pela indústria na produção de objetos em materiais diferentes dos usuais, como metais e cerâmica. (WANG, 2019)

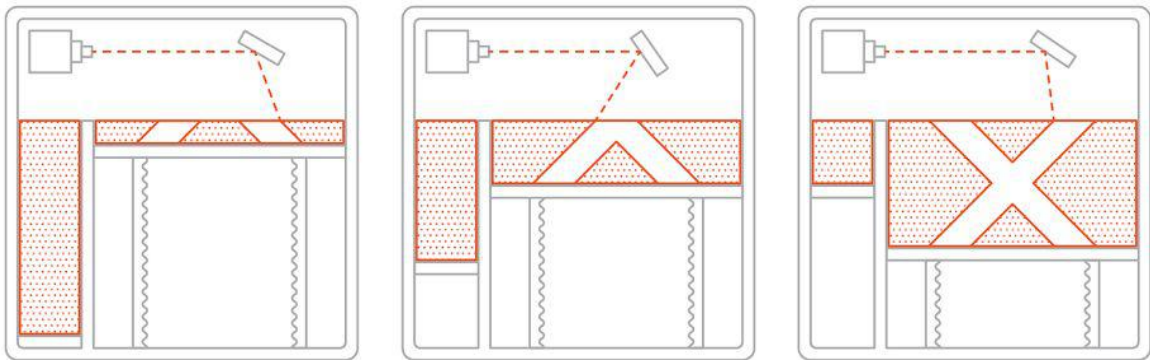


Figura 7 - Ilustração da técnica SLS

Fonte: (CORE77, 2017)



Figura 8 - Peça sendo retirada do recipiente após ser impressa utilizando a técnica SLS

Fonte:(BUILDPARTS, 2013)

3.2.2 Tipos de materiais na manufatura aditiva

Dentre os diversos materiais utilizados na manufatura aditiva, alguns se destacam pela sua facilidade de uso e custo reduzido, o que causa uma grande popularização dos mesmos, outros apresentam características únicas que interessam indústrias ou áreas mais específicas de estudo.

Entre os materiais mais populares, destaca-se o PLA (Poliácido Láctico), exemplificado na Figura 9, um polímero sintético termoplástico derivado de fontes renováveis, fazendo com que seja um plástico biodegradável (se dispuser de condições ideais) (ECYCLE, 2017). O ABS, plástico derivado do petróleo e com custo mais acessível dentre os demais. E o PETG, um termoplástico de polietileno glicol.

Estes materiais se diferenciam em propriedades mecânicas, preço e resistência. A comparação de suas características não é relevante para este estudo, portanto não será realizada nesta pesquisa.

Outros materiais utilizados em impressão 3D são ligas metálicas, polímeros, cerâmica, materiais pastosos, alimentos e suas combinações em forma híbrida. Estes materiais são utilizados com técnicas e finalidades diferentes, dependendo da área de atuação.

Exemplos de utilização relevantes são a utilização de impressão 3D com metal na indústria automobilística e aeroespacial, impressão de alimentos e peças comestíveis na indústria alimentícia, uso de materiais flexíveis na indústria têxtil para criação de itens de moda, vestimenta e calçados.



Figura 9 - Rolos de filamento PLA

Fonte:(3DERS, 2019)

3.2.3 Aplicações

As aplicações possíveis com a impressão 3D são limitadas apenas à imaginação do usuário. Com a possibilidade de utilização de variados materiais e técnicas diversas, é possível produzir uma infinidade de objetos com essa tecnologia.

Na indústria aeroespacial, essa tecnologia proporciona uma enorme liberdade de *design* para produção de peças, sendo possível realizar geometrias complexas e aprimoradas de maneira simples, gastando menos recursos e energia.

Já na indústria automobilística, atualmente já são fabricadas peças para automóveis em metal e materiais diferentes. Grandes empresas como Ford e BMW já utilizam a tecnologia em suas fábricas para criação de protótipos, peças de motores e ferramentas. (SREEHITHA, 2017).

E essa indústria não se limita às peças, em 2014, a empresa Local Motor criou o primeiro carro totalmente impresso em 3D, para mostrar o enorme potencial que esta tecnologia tem.

Outra área com grande atuação e utilização desta tecnologia é a construção civil e arquitetura. Casas impressas com grandes impressoras 3D já são uma realidade e mostram mais uma vez os grandes feitos da tecnologia.

Na área da saúde, a manufatura aditiva vem sendo bastante utilizada nos últimos anos em pesquisas científicas para criação de materiais biológicos, pesquisas para criação de medicamentos, impressão de ossos, cartilagens e tecidos. Existem muitas vantagens com o uso da impressão 3D na área da saúde, como por exemplo, a possibilidade de impressão de ossos e cartilagens, até mesmo de órgãos e tecidos para transplantes.

Nesta área da saúde, mais especificamente na engenharia de tecidos, o termo Bioimpressão 3D é muito utilizado, que se caracteriza pela união de impressão 3D com o prefixo “bio”, para dar a característica de um material biológico.

3.2.4 As 4 grandes etapas da impressão 3D

Para a realização de um projeto de impressão 3D, temos a necessidade de passar por 4 etapas importantes, que estão ilustradas em mais detalhes na Figura 10, são elas:

- Desenvolvimento da ideia do projeto:

Esta etapa nada mais é do que documentar o projeto a ser impresso, tirar a ideia da cabeça e descrevê-la em algum local para ser avaliada, refinada e melhorada de modo que se torne algo tangível.

- Modelagem da peça que será impressa:

Após a documentação da ideia, é necessária a criação do modelo 3D do projeto. Podem ser utilizados *softwares* convencionais de CAD, como AutoCAD e SketchUp para criação do modelo, bem como *softwares* de modelagem como Blender, Maya, Z Brush, Fusion 360, para citar alguns. Essa etapa é uma das mais importantes do processo, pois nela o modelador desenvolverá o modelo do projeto para ser impresso, o modelo 3D é uma prévia fiel do modelo final.

- Fatiamento do modelo 3D:

A etapa de fatiamento é onde todas as configurações são feitas, nela é utilizado um *software* de fatiamento, dentre eles podemos citar o Simplify3d, Cura e Slic3r, para analisar o modelo 3D da etapa anterior e fazer um fatiamento virtual da peça com todas as configurações que serão informadas para a impressora, tais como temperatura do bico de impressão, altura da camada de impressão, velocidade, temperatura da mesa e etc. Após isso, é gerado um arquivo chamado GCODE, contendo essas configurações, que será lido e interpretado pela impressora 3D. Nesta etapa, caso o fatiamento seja realizado de maneira incorreta ou sem as devidas configurações certas para o tipo de impressora, não será possível chegar a um resultado satisfatório.

- Impressão:

A última grande etapa consiste na transferência do arquivo GCODE para a impressora, que dependendo do modelo da impressora pode ser feito diretamente pela internet, por cartão de memória ou ligando diretamente o computador na impressora.

Feito isso, é necessário apenas iniciar a impressão e esperar a peça ficar pronta.

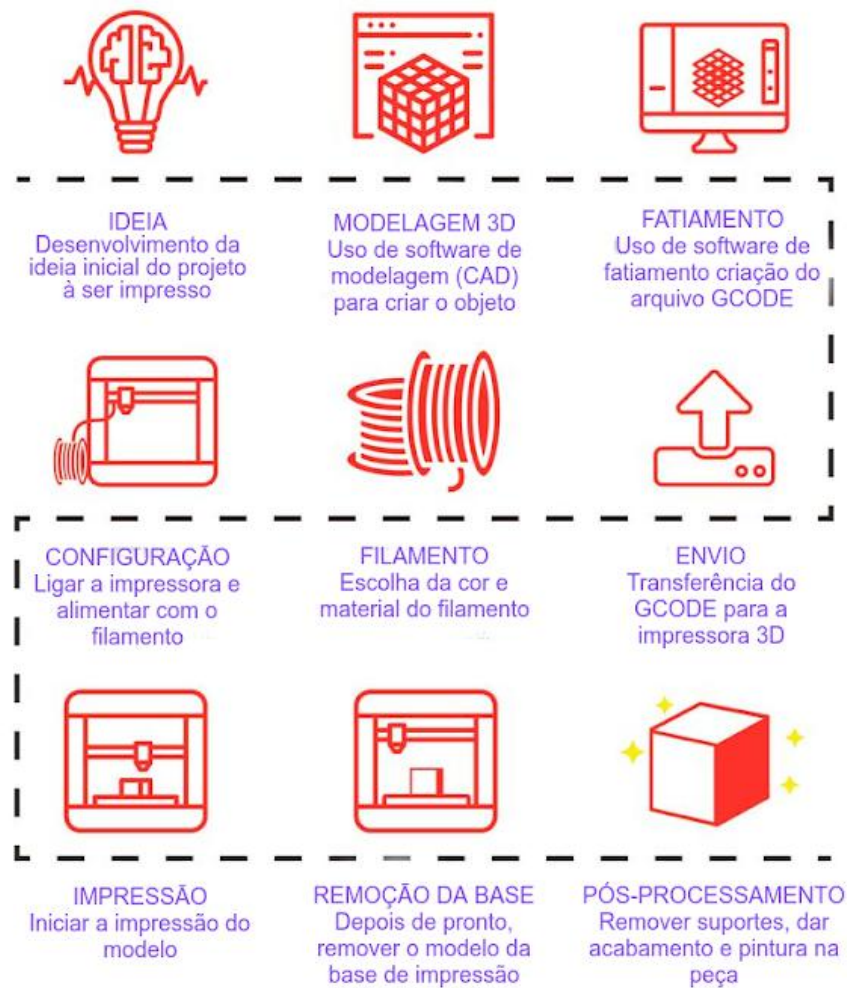


Figura 10 - Fluxo das etapas importantes de impressão 3D

Fonte: (THESTEMPEDIA, 2019). Traduzido pelos autores

3.3 BIOIMPRESSÃO 3D

A bioimpressão tridimensional (3D) é uma técnica em evolução que deve revolucionar o campo da medicina regenerativa. Como a doação de órgãos não atende às demandas de órgãos transplantáveis, é importante pensar em outra solução, que pode e muito provavelmente será fornecida por essa tecnologia (MIRONOV, 2018).

A bioimpressão pode ser definida pela fabricação aditiva de biomateriais utilizando de células e outros produtos biológicos produzindo modelos com heterogeneidade das propriedades físicas, da composição celular e da matriz extracelular (MEC). Essa abordagem inovadora é cada vez mais utilizada na biomedicina e tem potencial para projetar tecidos,

órgãos artificiais, além de criar protótipos funcionais para triagem de medicamentos, para pesquisa toxicológica e também transplante de tecidos e órgãos (HÖLZL, 2016)

A engenharia de tecidos e a medicina regenerativa obtiveram um crescimento sem precedentes na última década, levando o campo dos modelos de tecidos artificiais a uma revolução na medicina. Grandes progressos foram alcançados através do desenvolvimento de estratégias inovadoras de biomanufatura para padronização e organização de células e matriz extracelular (MEC) em três dimensões (3D) para criar construções funcionais de tecido. A bioimpressão surgiu como uma promissora tecnologia de biomanufatura em 3D, permitindo um controle preciso sobre a distribuição espacial e temporal das células e da MEC. (MIRONOV, 2018; ARSLAN-YILDIZ 2017).

3.3.1 Técnicas de bioimpressão

Essa recente tecnologia pode ser classificada de acordo com o seu princípio de funcionamento sendo dividido em três principais grupos como mostrado na Figura 11: Bioimpressão baseada em sinterização à laser (*laser assisted bioprinting*), em extrusão de material (*extrusion bioprinting*) e bioimpressão baseada em jato de biotinta (*inkjet bioprinting*), sendo esta última, a técnica mais promissora descrita na atualidade, pois com algumas alterações em impressoras 3D convencionais já é possível criar uma bioimpressora 3D (AVCI, et al. 2017).



Figura 11 - Diferentes técnicas de bioimpressão 3D

Fonte: Huseyin Avci et al. (2017), adaptado pelos autores

Impressão à jato de Biotintas

A bioimpressão por jato de biotinta foi a primeira abordagem a ser desenvolvida, a partir da modificação de impressoras comerciais a jato de tinta comuns e, como tal, se assemelha ao

mecanismo da impressão 3D convencional. A impressão a jato de tinta pode ser realizada como jato de tinta contínuo (JTC) ou deposição sob demanda (DSD). O primeiro produz um fio ininterrupto, enquanto o último deposita gotículas de tinta em posições coordenadas com precisão. A abordagem DSD é a única considerada adequada para *bioprinting*, uma vez que o método JTC requer uma tinta condutora, que acaba degradando o material biológico (TURK, 2018; MIRONOV, 2017).

Um cartucho de hidrogel é carregado com células e é impresso em gotículas precisamente alinhadas que são geradas a partir de um bico de impressão pelo controle rigoroso de um atuador térmico ou piezoelétrico como ilustrado na Figura 12.

No sistema térmico a temperatura do atuador pode exceder 200°C e mesmo com essa temperatura elevada, as células apenas sofrem pequenas mudanças temporárias de temperatura durante o processo, o sistema térmico é dominante na bioimpressão à jato de Biotintas (McIlwraith, 2016).

Atualmente, os bicos de impressão a jato de biotinta tendem a empregar o sistema Micro-Eleto-Mecânico(SMEM) que empenham-se para imprimir materiais altamente viscosos ou com altas concentrações de células (IRVINE, 2016).

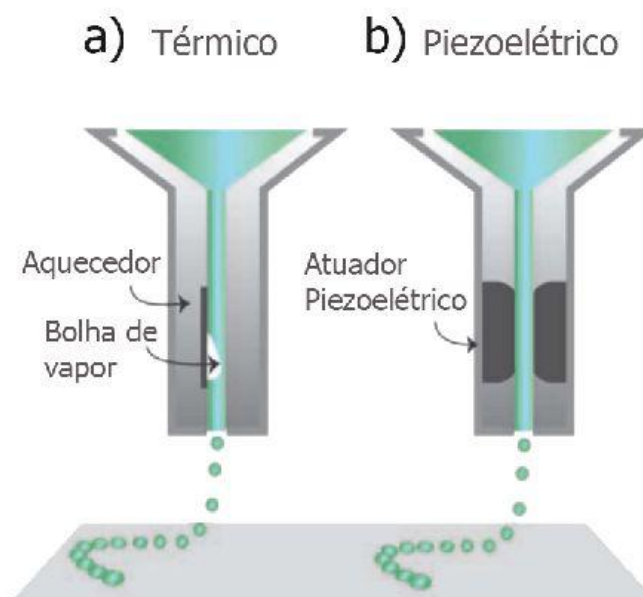


Figura 12 - Uma representação da bioimpressão a jato de Biotinta, onde a partir de um reservatório, gotas de biotinta são formadas pela ação de um atuador piezoelétrico(b) ou térmico(a) e são depositadas em uma placa de coleta.

Fonte: Huseyin Avci et al. (2017), adaptado pelos autores

Bioimpressão baseada em microextrusão de material

A bioimpressão baseada em extrusão de material é a abordagem mais promissora dentre as três citadas neste estudo, pois ela permite a fabricação de estruturas organizadas de tamanhos clinicamente relevantes dentro de um período de tempo satisfatório. (DERNOWSEK, 2018).

Essa técnica produz um fio contínuo de filamento de biotinta a partir de uma seringa. O hidrogel pode ser forçado a partir da seringa de três maneiras distintas, usando pressão de ar (pneumática), pistão ou parafuso mecânico, mostrado na Figura 13. A seringa é conectada ao braço de impressão, movendo na direção z-y sobre uma placa coletora que move o eixo x, para que o hidrogel possa ser depositado nos padrões 3D (IRVINE, 2016).

Diversos tipos de polímeros de hidrogel biocompatíveis podem ser utilizados como biotinta para bioimpressão baseada em extrusão e tem a capacidade de fornecer uma concentração celular relativamente alta. O aparelho pneumático é melhor utilizado na impressão de biotintas mais viscosas, porém eles sofrem com um atraso no controle ao tentar interromper o fluxo de biotinta do bico. O sistema de pistão e o de parafuso mecânico oferecem melhor controle sobre o fluxo da biotinta oferecendo um padrão melhorado; no entanto, para impressão em células-tronco, a extrusão por parafuso mecânico pode produzir grandes quedas de pressão ao longo do bico, esta queda está associada à deformação e apoptose das células encapsuladas por biotinta (MIRONOV, 2017).

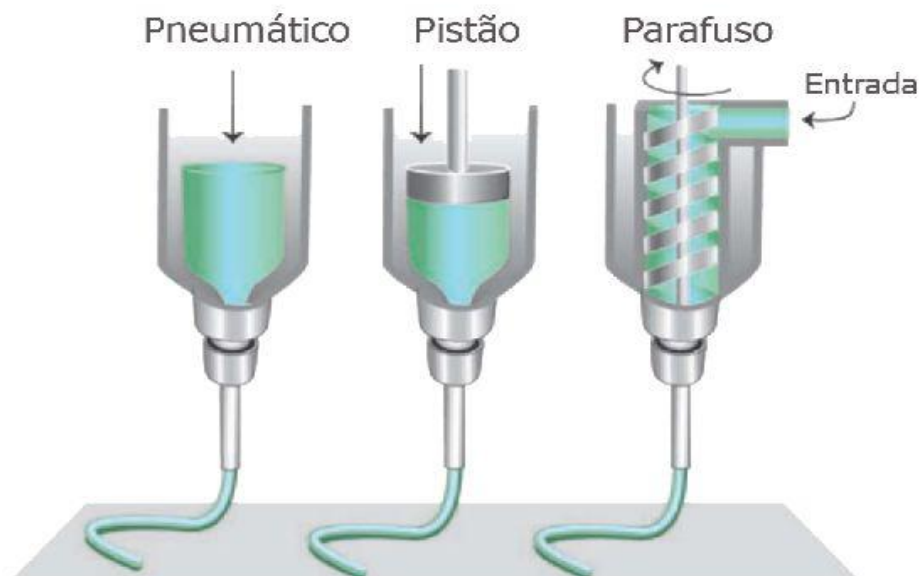


Figura 13 - Formas de microextrusão de material

Fonte: Huseyin Avci et al. (2017), adaptado pelos autores

Bioimpressão baseada em sinterização à laser

Essa abordagem foi desenvolvida a partir de tecnologias de gravação direta a *laser* e transferência induzida a *laser*. A bioimpressão assistida por laser concentra os pulsos de *laser* na lâmina doadora, criando assim alta pressão para impulsionar gotículas de hidrogel carregado de células para a lâmina coletora, demonstrado na Figura 14. Ao contrário da impressão a jato de tinta, a bioimpressão assistida por laser não requer um bico de distribuição. Isso reduz o estresse de cisalhamento sofrido pelas células-tronco durante a deposição e também remove a ocorrência de entupimento dos bicos.

Este método é considerado como tendo boas propriedades de impressão celular, pois é capaz de usar biotintas mais viscosas e imprimir altas densidades de células com boa viabilidade pós-deposição. Além disso, a resolução pode ser tão baixa quanto 10 μm . No entanto, a técnica é considerada a mais cara e complexa.(IRVINE, 2016).

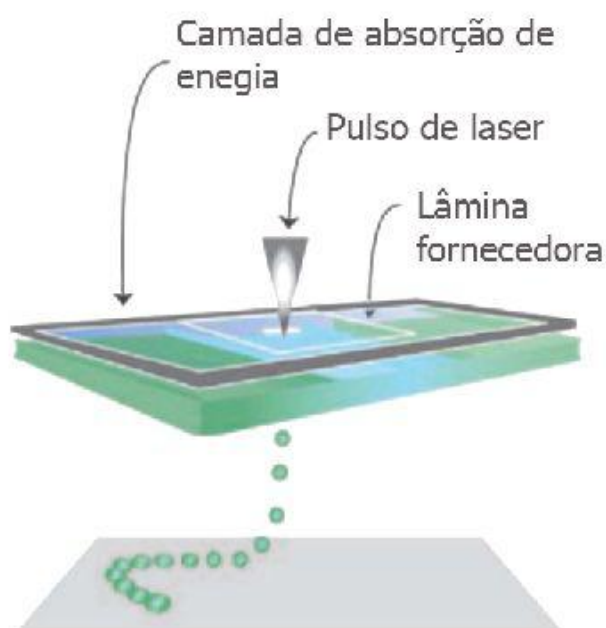


Figura 14 - Diagrama esquemático da bioimpressão baseada em sinterização à laser

Fonte: Huseyin Avci et al. (2017), adaptado pelos autores

Cada modelo de tecnologia possui seus pontos positivos e negativos dependendo de suas características, pode variar na densidade das células, resolução, velocidade de impressão e custo, todas demonstradas na Tabela 3.

Tabela 3 - Tabela comparativa das modalidades de bioimpressão 3D

	Jato de biotinta	Microextrusão	Sinterização à laser
Densidade de células	Baixo, 10^6 células/ml	Sem limitação	Médio 10^8 células/ml
Resolução	Alta	Média	Alta
Velocidade de impressão	Rápido	Devagar	Moderado
Custo	Baixo	Médio	Alto

Fonte: Ilze Donderwinkel. (2017), adaptado pelos autores

3.3.2 As 4 grandes etapas da bioimpressão 3D

A tecnologia de bioimpressão possui etapas semelhantes à da impressão 3D, podendo ser resumida basicamente em quatro etapas (MIRONOV, 2017):

- Pré-processamento

Antes de iniciar o processo de bioimpressão é necessário a seleção de células, a cultura celular e a criação de esferóides teciduais (aglomerados celulares) que podem ser produzidos utilizando diversas técnicas diferentes, uma delas é o Hanging Drop, ilustrado na Figura 15. Muitos artigos exemplificam a utilização de imagens de tomografia ou ressonância para a bioimpressão, mas é necessário um tempo significativo de estudos sobre a estrutura, a função dos tecidos vivos e a composição de cada tecido/órgão. A complexidade, a dinamicidade das células e a remodelagem biológica dos seus componentes é crucial para um projeto funcional do tecido.

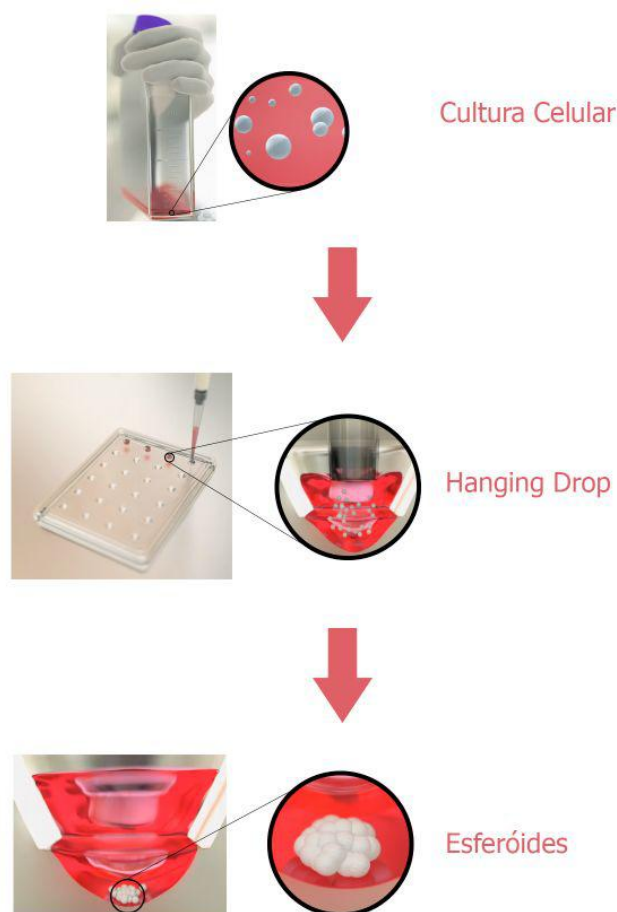


Figura 15 - Formação de esferóides pelo método Hanging Drop

Fonte: Otávio Henrique Junqueira Amorim, adaptado pelos autores

Grandes esforços têm se concentrado na combinação de materiais biodegradáveis e células, devido à limitação encontrada com o uso de estruturas sólidas convencionais e nas abordagens utilizando apenas células não dependentes de *scaffolds* sólidos (*solid free scaffolds*).

- Modelagem e fatiamento

Após a cultura celular é necessário modelar e fatiar o arquivo para bioimpressão, como essa área é nova, muitos dispositivos, *softwares* e métodos foram adaptados pois não foram devidamente criados para a bioimpressão. Assim, a integração e a interoperabilidade de dados, arquivos e sistemas tornam-se um desafio enorme.

O uso de métodos adaptados é rotina e atualmente o modelo mais frequentemente utilizado segue o seguinte caminho: é criado o modelo do tecido/órgão, depois é usado um software para fazer o fatiamento virtual da peça em camadas, mas como não existem opções criadas especificamente para a bioimpressão 3D, são utilizados softwares da impressão 3D

convencional adaptados e alguns programas adicionais para tornar possível a bioimpressão, porém é desperdiçado muito tempo fazendo conversões entre estes programas.

Após ser gerado o projeto fatiado no formato GCODE, é utilizado o Gcode Editor para editar esse arquivo, retirando informações desnecessárias para bioimpressão, ao salvar, este arquivo é convertido para o formato .ptl, no qual é utilizado mais um *software*, desta vez o PetriPrinter que contém modelos prontos de placas de petri (recipiente que auxilia na microbiologia), onde o tecido/órgão é impresso e assim o BioCAD, ou seja, o arquivo com a organização estrutural dos componentes do tecido é finalmente gerado.

- Processamento ou bioimpressão 3D

Uma segunda etapa chamada de processamento ou bioimpressão 3D, utiliza componentes como, células, esferóides, materiais sintéticos mesclados aos biomateriais, hidrogéis e biomoléculas, que são depositados camada, por camada formando assim o tecido/órgão desejado, como pode ser visto na Figura 16.

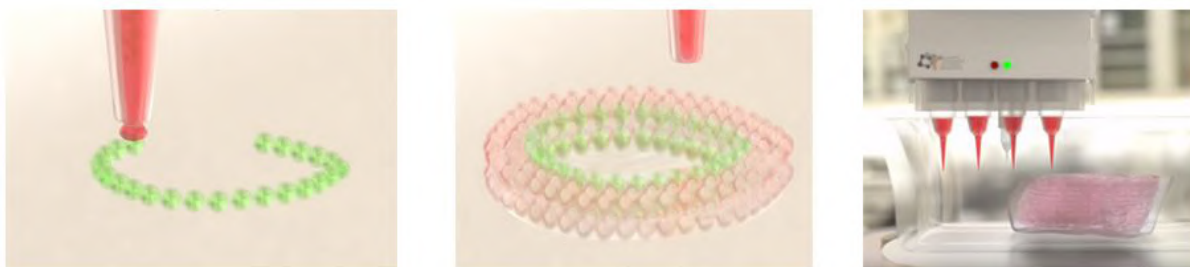


Figura 16 - Processo de bioimpressão 3D

Fonte: Otávio Henrique Junqueira Amorim, adaptado pelos autores

-Pós-processamento ou maturação do tecido bioimpresso.

Período de maturação do tecido bioimpresso. Essa última fase necessita de sistemas fechados e controlados para o amadurecimento deste tecido, são utilizados equipamentos chamados biorreatores, ilustrado na Figura 17, que permitem a remoção de excreções e a entrada de fluidos, essenciais para o desenvolvimento do tecido bioimpresso (DERNOWSEK, 2017).



Figura 17 - Biorreator de perfusão virtual
 Fonte: www.inctregenera.org.br

O equipamento utilizado para a biofabricação de tecidos, e futuramente grandes órgãos, é chamado de bioimpressora, exibida na Figura 18. Este equipamento é análogo à uma impressora 3D. A principal diferença é o material biológico que é utilizado, ao invés do plástico comum.

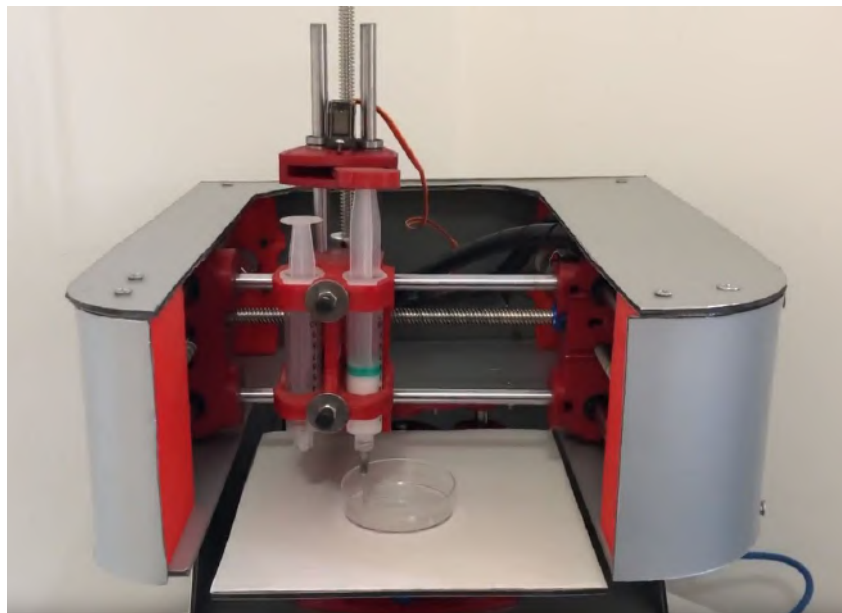


Figura 18 - BioEdPrinter - Bioimpressora básica da startup BioEdTech
 Fonte: Janaina Dernowsek, adaptado pelos autores

Um ambiente ideal para melhorar e facilitar a logística de fabricação de órgãos seria uma linha de biofabricação integrada, como ilustrada na Figura 19.

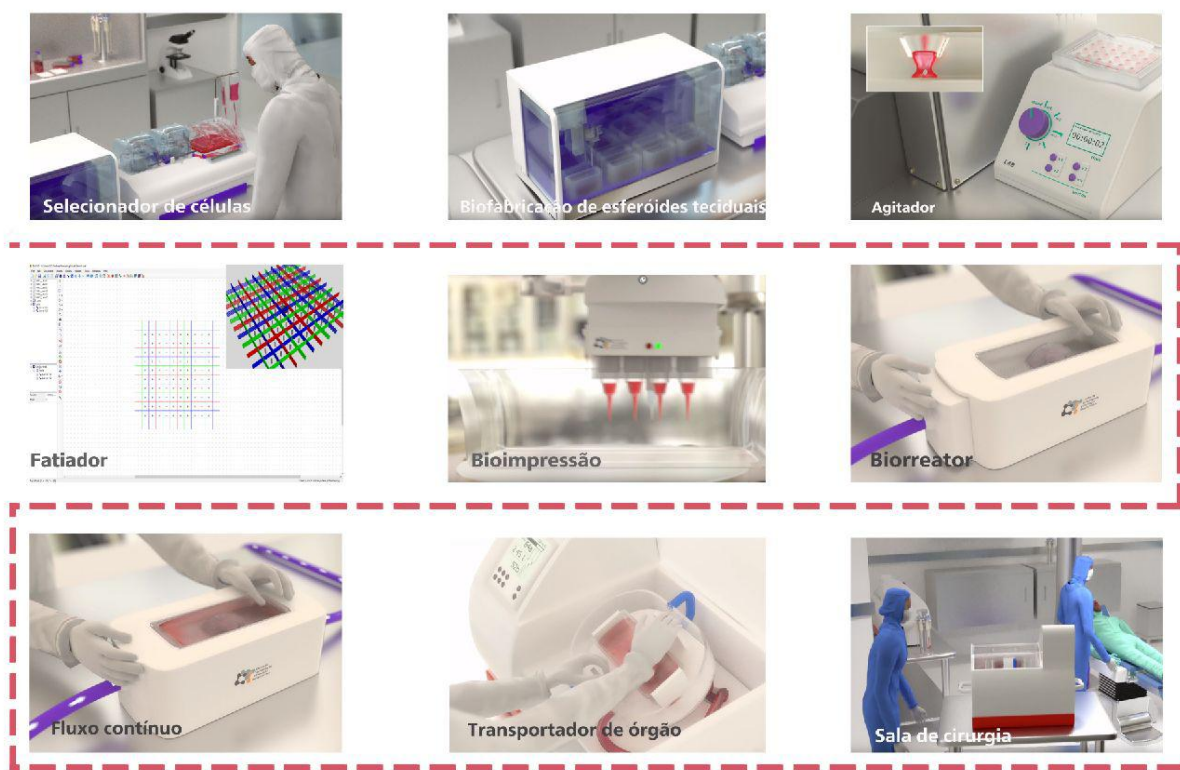


Figura 19 - Fluxo das etapas importantes de bioimpressão 3D

Fonte: Otávio Henrique Junqueira Amorim, adaptado pelos autores

3.3.3 Softwares de fatiamento para bioimpressão 3D

Atualmente não existem *softwares open source* dedicados apenas à bioimpressão 3D, apenas existem biofatiadores que estão atrelados à compra de bioimpressoras, ou seja, são soluções fechadas. Dentre elas estão o **BIOCAM** e o **Allevi** os dois prometem ser um fatiador de fácil utilização e multifuncional.

Mesmo com essa escassez de *softwares* dedicados apenas à bioimpressão 3D, alguns fatiadores da impressão 3D tradicional são adaptados e utilizados para bioimpressão 3D.

Por fim, não haver programas ou métodos computacionais criados especificamente para o desenvolvimento de tecidos biológicos complexos não impede a criação e o desenvolvimento da biotecnologia, apenas dificulta o amadurecimento da técnica e nos incentiva a desenvolver um facilitador para que essa tecnologia seja pioneira no Brasil.

3.3.4 Desafios da bioimpressão

Os desafios presentes na bioimpressão de tecidos e órgãos são inúmeros. Podemos dividir esses desafios em tecnológicos, biológicos, químicos, físicos, de materiais, éticos e de regulamentação.

Os desafios tecnológicos são complexos e precisam ser desenvolvidos, pois a área é nova e muitos dispositivos, *softwares* e métodos foram adaptados e não foram devidamente criados para a bioimpressão. Assim, a integração e a interoperabilidade de arquivos e sistemas tornam-se um grande desafio. O uso de métodos adaptados da engenharia é rotina, como por exemplo o uso de *softwares* da engenharia para a modelagem do BioCAD, ou seja, do projeto do tecido que será biofabricado.

Outro fator desafiante na bioimpressão 3D de grandes órgãos é que as células são sensíveis ao estresse mecânico, que pode se tornar significativo quando os hidrogéis (*bioinks*), carregados de células são forçados através de um pequeno orifício, como o bico de impressão. Assim, os *bioinks* geralmente são menos viscosos, para garantir que as células possam ser depositadas com alta viabilidade. Para garantir que estruturas grandes sejam adequadamente suportadas, cada camada deve manter sua forma quando impressa. No entanto, com *bioinks* de baixa viscosidade, essa é uma tarefa altamente difícil, beirando a impossibilidade, pois eles fluem e se espalham rapidamente após a ejeção do bico, sem manter o formato que deveriam.

4 PROPOSTA DE MELHORIA NO *SOFTWARE* DE FATIAMENTO

4.1 BIOSLIC3R

O BioSlic3r, nome dado à melhoria desenvolvida nesta pesquisa para contribuir com o *software* de fatiamento Slic3r e assim passar a ter suporte para bioimpressão 3D, foi desenvolvido com base em uma sugestão da pesquisadora PhD Janaína Dernowsek, para integrar dois *softwares open source* utilizados na bioimpressão 3D, são eles o já citado fatiador Slic3r e o *software* PetriPrinter, seu menu pode ser visto na Figura 20.

O PetriPrinter é um *software* gerador de GCODE, arquivo lido pela impressora 3D, que facilita o alinhamento da bioimpressão 3D em placas de Petri, para cultura de material biológico, como ilustrado na Figura 21.

Em nossa pesquisa, analisamos a possibilidade de integrar os dois *softwares*, desenvolvendo uma maneira de converter o GCODE gerado pelo Slic3r, no arquivo lido pelo PetriPrinter diretamente de um menu no fatiador Slic3r, sem que haja necessidade de usar *softwares* terceiros, como é o caso atual, que para convertermos um GCODE para a extensão aceita pelo PetriPrinter, precisamos utilizar o *software* gCode Editor.

Esta contribuição elimina uma etapa do processo de bioimpressão 3D deixando assim mais simples e intuitivo.

O nosso objetivo com isso é tornar o processo de bioimpressão 3D o mais intuitivo possível, para que profissionais da área médica e biológica, que não tem conhecimento na área de tecnologia da informação possam ser apresentados à bioimpressão 3D e aprendam com mais facilidade, para que essa tecnologia seja mais disseminada e utilizada por profissionais que claramente serão beneficiados com o uso da mesma.

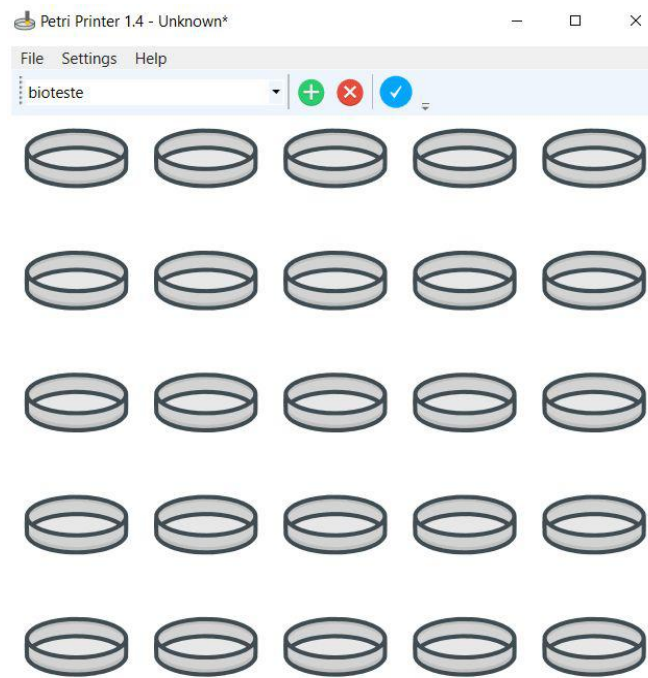


Figura 20 - Tela do software PetriPrinter

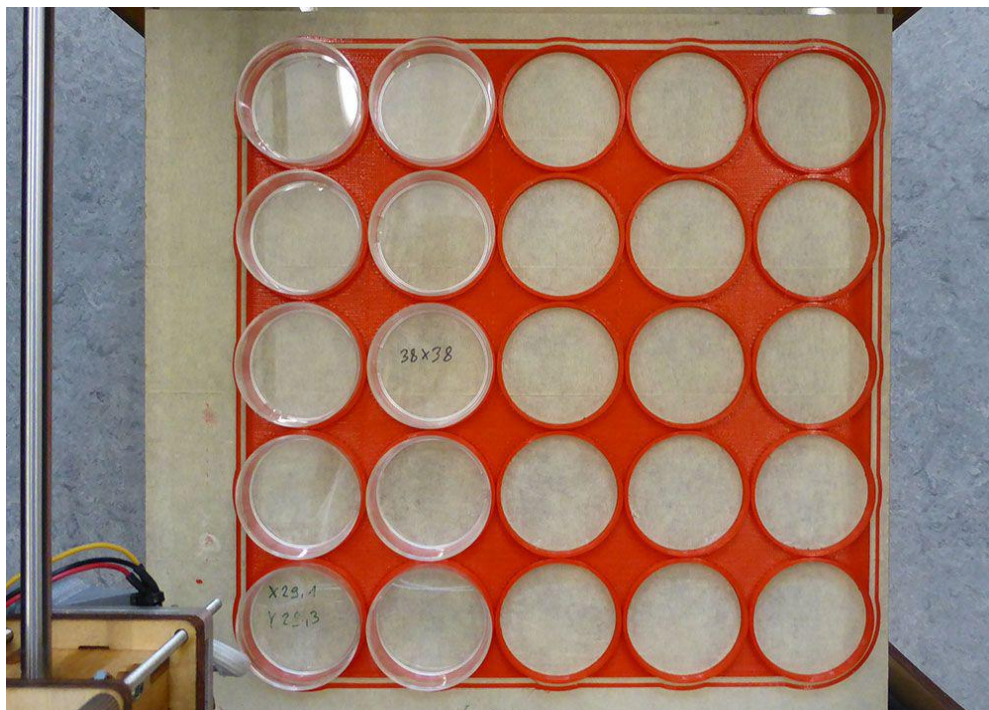


Figura 21 - Guia para placas de Petri desenvolvido com o software PetriPrinter

Fonte: www.petriprinter.elte.hu/main/petriprinter

4.2 FLUXOS DE CONTRIBUIÇÃO DO BIOSLIC3R

O fluxo atual do processo de bioimpressão utilizando o Slic3r está exibido na Figura 22.

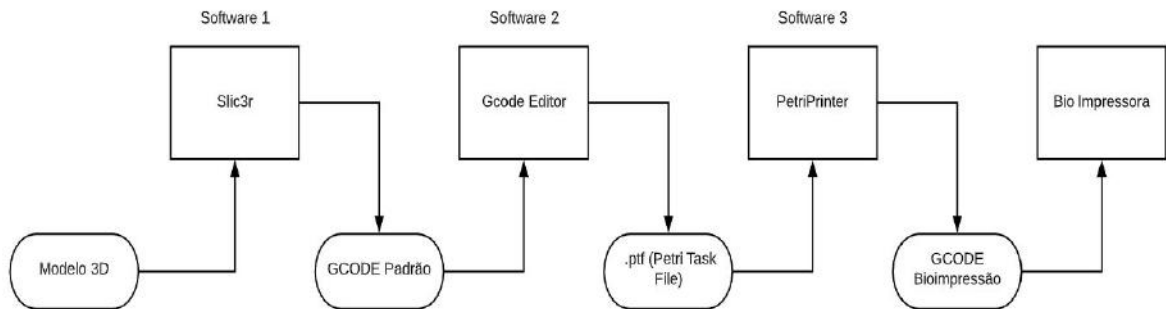


Figura 22 - Fluxo atual de bioimpressão

Com a contribuição desenvolvida nesta pesquisa, chegamos no fluxo atual ilustrado na Figura 23.

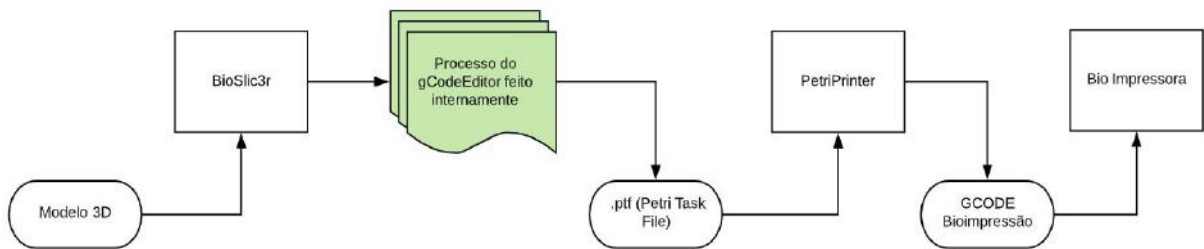


Figura 23 - Fluxo alcançado com o desenvolvimento inicial do BioSlic3r

Dada a continuação no desenvolvimento, a previsão para o cenário futuro é chegar no fluxo ilustrado na Figura 24.

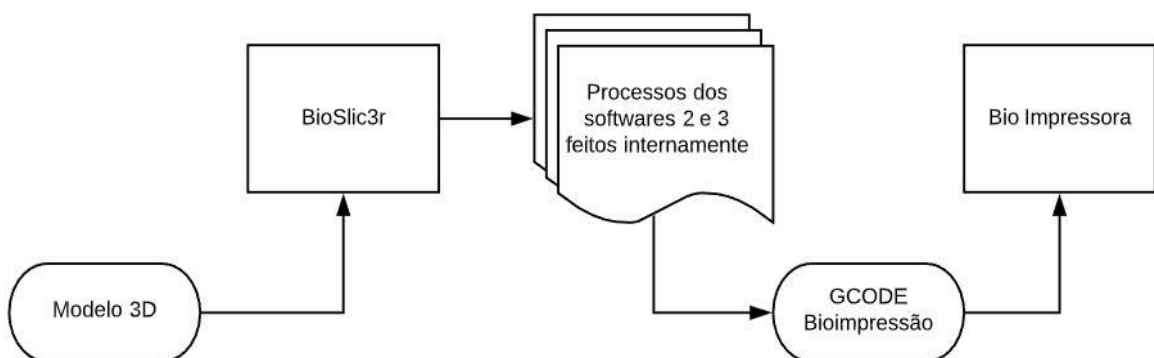


Figura 24 - Fluxo esperado para trabalhos futuros com etapas integradas no fatiador Slic3r

5 DESENVOLVIMENTO DO ALGORITMO

5.1 AMBIENTE DE DESENVOLVIMENTO

Antes de discutir qualquer assunto técnico, ou até mesmo discutir basicamente o tópico em questão neste estudo, devemos discutir sobre onde e como instalaremos a solução. No caso do BioSlic3r, os ambientes de desenvolvimento e teste a princípio são bem definidos, como visto na Tabela 4, porém não são necessariamente excludentes quanto às outras ferramentas e ambientes também popularmente utilizados, como o sistema operacional Windows. A definição do ambiente em si é baseada em experiência prévia dos autores envolvidos com a ferramenta e testes feitos, de fato, pelos desenvolvedores.

Tabela 4 - Ambiente de Teste e Desenvolvimento

Tecnologias	
Fedora Linux	Versão 31
Memória	16GB
Processador	Intel Core i7-8650U
GNOME	Version 3.34.2
Tipo de SO	64-bit
Disco SSD	256GB

Fonte: Desenvolvido pelos autores

O algoritmo foi implementado em um ambiente Linux, com ferramentas atuais de desenvolvimento como o **Git, para versionamento e o Atom**, IDE para escrita e gerenciamento dos diretórios do código fonte. O sistema operacional Linux é um ambiente bastante favorável para implementação de novas funcionalidades e melhorias por ser *open source* e de fácil gerenciamento. Porém, é de conhecimento geral de que boa parte dos usuários de computadores usam um ambiente Windows. De fato, existe a preocupação quanto à usabilidade da ferramenta para que a experiência realmente seja melhorada, porém o sistema Windows se mostrou menos eficiente em termos de atualização de dependências, instalação do ambiente de desenvolvimento e configuração das ferramentas necessárias. Esse assunto, em particular, será abordado mais à frente como forma de futuras contribuições.

5.2 DESCRIÇÃO DO DESENVOLVIMENTO

A integração em si é bem simples de ser explicada superficialmente, porém os processos envolvidos são bem complexos em termos computacionais. Envolvem duas ferramentas: gCodeEditor e Slic3r, que são desenvolvidas em linguagens diferentes, C# e C++ respectivamente. O primeiro desafio é conseguir “traduzir” não só toda a lógica do algoritmo, mas também as funções utilizadas, já que muito provavelmente, nem sempre será possível utilizar os mesmos recursos em ambientes e linguagens diferentes. A idéia é justamente trazer a funcionalidade do gCodeEditor de gerar o arquivo .ptf, arquivo lido pelo PetriPrinter, pro Slic3r e tentar reduzir a complexidade de toda a operação.

Em termos práticos, apesar de conseguirmos sintetizar o objetivo do estudo em algumas palavras, as dificuldades são reais e posteriormente podem se tornar trabalhos futuros. O Slic3r em si já uma aplicação robusta, apesar de bem estruturada e documentada, que ~~envolve~~ exige certo tipo de expertise e estudo para que possamos contribuir em cima do *software* existente. Nos subtópicos abaixo, serão listados alguns dos principais problemas enfrentados e como resolvemos.

5.2.1 Função Serialize

Como exemplo da dificuldade de “tradução” de uma linguagem à outra, podemos citar a função serialize. O gCodeEditor usa na sua parte final da edição e processamento do arquivo de saída, uma função chamada Serialize, que codifica o arquivo de saída em formato binário, de forma que ele é entendido e processado pelo PetriPrinter posteriormente. Essa função faz parte de uma biblioteca local presente no projeto do PetriPrinter chamada PetriIO. Mais à fundo, em baixo nível, essa é uma função de uma *runtime* do sistema. Essa função é crucial pois é dessa forma que o PetriPrinter trabalha: ao receber um código binário, o *software* está pronto para ler as informações do arquivo e processá-las para a impressora recebê-lo. Ao trazermos essa funcionalidade para o c++/perl, devemos da mesma forma importar a *runtime* padrão do sistema que nos possibilita usar a função de serializar e desserializar o código. Podemos utilizar uma das bibliotecas mais utilizadas chamada Boost, que é nativa e facilita o uso das funções com os arquivos. Essa é a forma mais fácil e mais trivial de conseguirmos integrar a funcionalidade à ferramenta.

5.2.2 Alinhamento da estrutura “backbone” do Slic3r

O Slic3r é em sua maior parte feito em C++ e Perl, duas linguagens comumente utilizadas no mundo da tecnologia. Todo o projeto é bem estruturado, com os diretórios definidos da forma mostrada na Figura 25.

```

package/ : the scripts used for packaging the executables
src : the C++ source of the slic3r executable and the CMake definition file for compiling it
src/GUI : The C++ GUI.
src/test : New test suite for libslic3r and the GUI. Implemented with Catch2
t/ : the test suite (deprecated)
utils/ : various useful scripts
xs/src/libslic3r/ : C++ sources for libslic3r
xs/t/ : test suite for libslic3r (deprecated)
xs/xsp/ : bindings for calling libslic3r from Perl (XS) (deprecated)

```

Figura 25 - Explicação dos diretórios do projeto Slic3r pela documentação oficial

Ao incluir uma nova funcionalidade e promover a integração das ferramentas, foi necessário entender e criar uma estratégia onde toda a lógica seria inserida no código atual. Dependendo de como decidimos a estrutura de dados correta, as funções e variáveis, as bibliotecas e dependências, podemos estar agindo diretamente na complexidade e bom funcionamento da ferramenta. Dessa forma, a diretiva a ser seguida foi sempre a de **descomplicar, porém não perder em eficiência.**

Em outras palavras, o gCodeEditor possui algumas classes que, conjuntamente, fazem toda a operação necessária para processar o arquivo .gcode, fazer as alterações necessárias e alterá-lo para .ptf. Agimos diretamente nelas para aproveitar o código existente, que funciona bem, computacionalmente falando. Dentre elas, existem duas principais, ilustradas nas Figuras 26 e 27 a seguir.

```

MainViewModel.cs
1  using GCodeAPI;
2  using gCodeEditor.Model;
3  using gCodeEditor.View;
4  using Microsoft.Win32;
5  using System;
6  using System.Collections.Generic;
7  using System.Collections.ObjectModel;
8  using System.ComponentModel;
9  using System.Linq;
10 using System.Text;
11 using System.Windows;
12
13 namespace gCodeEditor.ViewModel
14 {
15     public class MainViewModel : INotifyPropertyChanged
16     {
17         #region buttons
18         public BasicVezerloBtn CreateNewBtn { get; private set; }
19         public BasicVezerloBtn OpenBtn { get; private set; }
20         public BasicVezerloBtn SaveBtn { get; private set; }
21         public BasicVezerloBtn SaveAsBtn { get; private set; }
22         public BasicVezerloBtn ImportBtn { get; private set; }
23         public BasicVezerloBtn ShowGridSettingsBtn { get; private set; }
24         public BasicVezerloBtn ShowPrinterSettingsBtn { get; private set; }
25         public BasicVezerloBtn showHelpBtn { get; private set; }
26         public BasicVezerloBtn showAboutBtn { get; private set; }
27         public BasicVezerloBtn AddBtn { get; private set; }
28         public BasicVezerloBtn RemoveBtn { get; private set; }
29         public BasicVezerloBtn GenerateBtn { get; private set; }
30
31         public BasicVezerloBtn Up1 { get; private set; }
32         public BasicVezerloBtn Up10 { get; private set; }
33         public BasicVezerloBtn Up100 { get; private set; }
34         public BasicVezerloBtn UpZ { get; private set; }
35
36         public BasicVezerloBtn Down1 { get; private set; }
37         public BasicVezerloBtn Down10 { get; private set; }
38         public BasicVezerloBtn Down100 { get; private set; }
39         public BasicVezerloBtn DownZ { get; private set; }
40     }

```

Figura 26 - Classe MainViewModel

```

IOTools.cs
1  using System;
2  using System.Collections.Generic;
3  using System.IO;
4  using System.Linq;
5  using System.Runtime.Serialization.Formatters.Binary;
6  using System.Text;
7
8  namespace gCodeEditor.Model
9  {
10     public class IOTools
11     {
12         public static void Save(String path, object obj)
13         {
14             var formatter = new BinaryFormatter();
15             try
16             {
17                 using (var stream = new FileStream(path, FileMode.Create, FileAccess.Write, FileShare.None))
18                     formatter.Serialize(stream, obj);
19             }
20             catch (Exception e) { Console.WriteLine(e.Message); Console.WriteLine(e.StackTrace); }
21         }
22
23         public static object Open(String path)
24         {
25             var formatter = new BinaryFormatter();
26             try
27             {
28                 using (Stream stream = new FileStream(path, FileMode.Open, FileAccess.Read, FileShare.Read))
29                     return formatter.Deserialize(stream);
30             }
31             catch (Exception e) { Console.WriteLine(e.Message); Console.WriteLine(e.StackTrace); }
32             return null;
33         }
34
35         public static T Open<T>(String path)
36         {
37             var formatter = new BinaryFormatter();
38             try
39             {

```

Figura 27 - Classe IOTools

Essas classes reúnem boa parte das funções e da lógica principal do programa que é editar e exportar o até então arquivo `.gcode` para arquivo `.ptf`. Apesar de já existir um procedimento proveniente do *software* `gCodeEditor`, no qual queremos trazer para o `Slic3r`, claramente não se trata de simplesmente copiar toda a lógica e colar nos arquivos existentes do projeto atual. É preciso que tanto as funções quanto às variáveis do código estejam acopladas num mesmo escopo.

Para o nosso caso, haveria a possibilidade de criarmos uma nova classe que pudesse ser um elo entre as classes principais, para que as variáveis e funções pudessem se comunicar entre si, e as operações fossem bem executadas. Essa é uma alternativa classificada como *design pattern*, que agiria basicamente como superclasse, em que todas as outras classes estenderiam dela. Não nos pareceu tão entusiasmante já que isso alteraria a estrutura existente do projeto, e isso não foi considerado ideal para que futuras contribuições de terceiros sejam facilitadas.

De forma antagônica, buscamos reduzir a complexidade e também, efetivamente, o código. O trabalho de planejamento do código em si é mais árduo, já que precisamos pensar em problemas que seriam resolvidos com a inclusão da *design pattern*, mas dessa forma teremos ganhos em não aumentar o número de arquivos no projeto e na utilização do escopo atual existente do projeto.

5.3 TESTES, RESULTADOS E AVALIAÇÃO

O *software* alcançado hoje é funcional, com a nova funcionalidade sugerida bem implementada, com um resultado satisfatório alcançado e inclusive possibilitando novas colaborações. O código está disponível no github: github.com/levenhagen/Slic3r.

O programa foi gerado com as diretivas de configuração e dependências do próprio repositório original do `Slic3r`, já que o nosso projeto foi elaborado com base no anterior existente, através de uma cópia do repositório original. Dessa forma, mantemos versionadas todas as alterações e contribuições feitas anteriormente. O código original está disponível no github: github.com/slic3r/Slic3r.

Como falado na seção anterior, para chegarmos à esse resultado, o desenvolvimento da funcionalidade, que de fato é uma integração à ferramenta `Slic3r`, exigiu uma estratégia prévia para que o fluxo de execução ocorra de forma correta. Tecnicamente, essa integração pode ocorrer de várias formas: um número variado de classes pode ser criado, diferentes bibliotecas

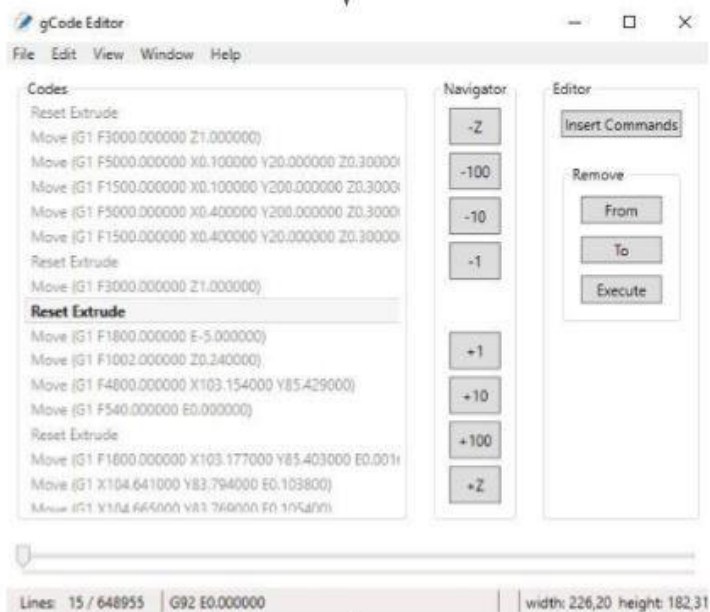
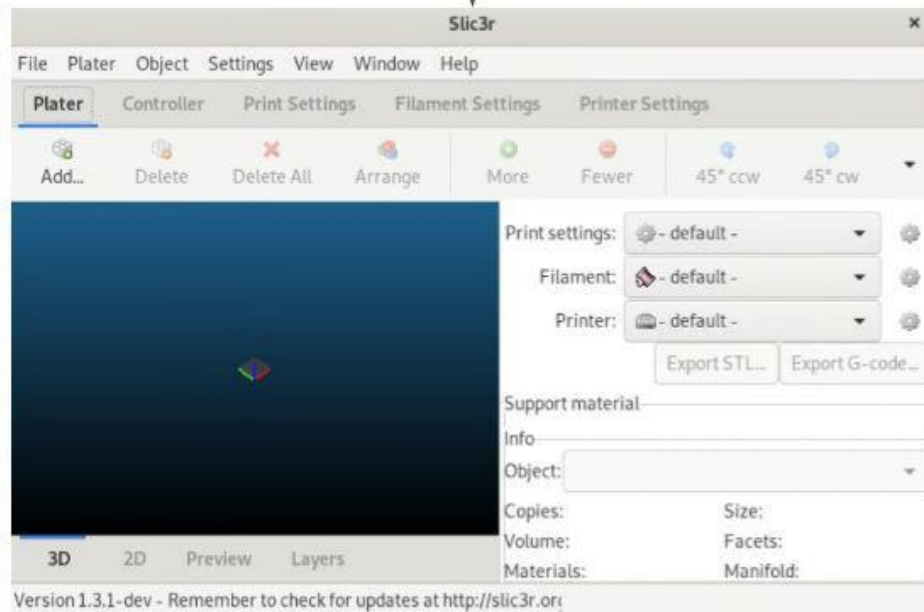
auxiliares, diferentes abordagens e estruturas de dados podem ser utilizados, a própria lógica do algoritmo pode ser diferente em alguns casos, etc.

Para o nosso objetivo, procuramos seguir a idéia de que o algoritmo deveria ser enxuto, facilmente legível, robusto e correto em termos de lógica de código. As boas práticas de código e desenvolvimento deveriam ser seguidas, como uma adoção à cultura *open source*. Queremos que a contribuição seja útil e aberta ao fácil entendimento bem como para novas contribuições, que é exatamente a ideia base da comunidade *open source*.

Para garantir o bom funcionamento do software desenvolvido, foram executadas boas práticas de testes de desenvolvimento, incluindo testes unitários, com o framework `CppUnit`, que garantem a execução correta das funções implementadas.

Ao final, considerando mudanças visuais e de interface, mostramos o fluxo de bioimpressão antes da modificação, exibido na Figura 28, e após o desenvolvimento da nova funcionalidade, com um novo menu no fatiador chamado de BioSlic3r, que hoje disponibiliza um espaço para recomendações ao usar o Slic3r para bioimpressão, bem como a principal funcionalidade adicionada que resumidamente é a de exportar o arquivo de saída com formato .ptf. e sem a necessidade do uso do gCodeEditor, ilustrado na Figura 29.

Modelo 3D



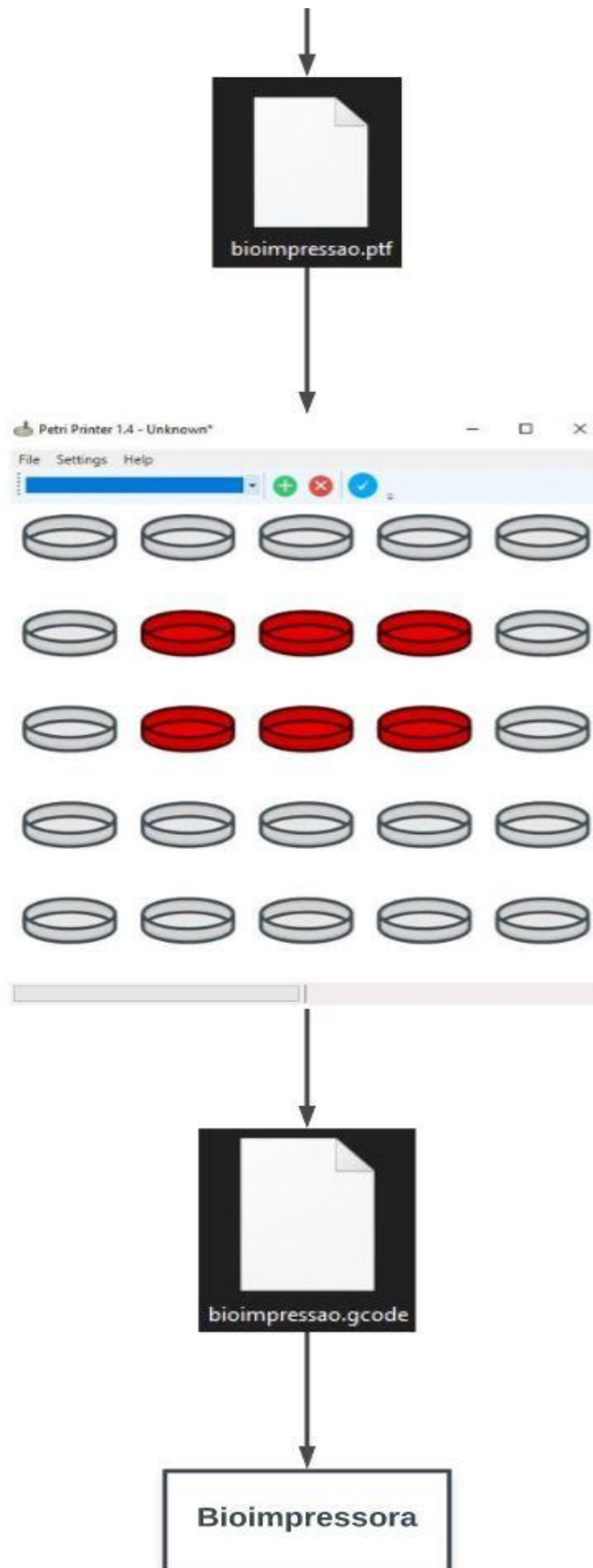


Figura 28 - Fluxo antigo de bioimpressão utilizando o Slic3r, gCodeEditor e PetriPrinter

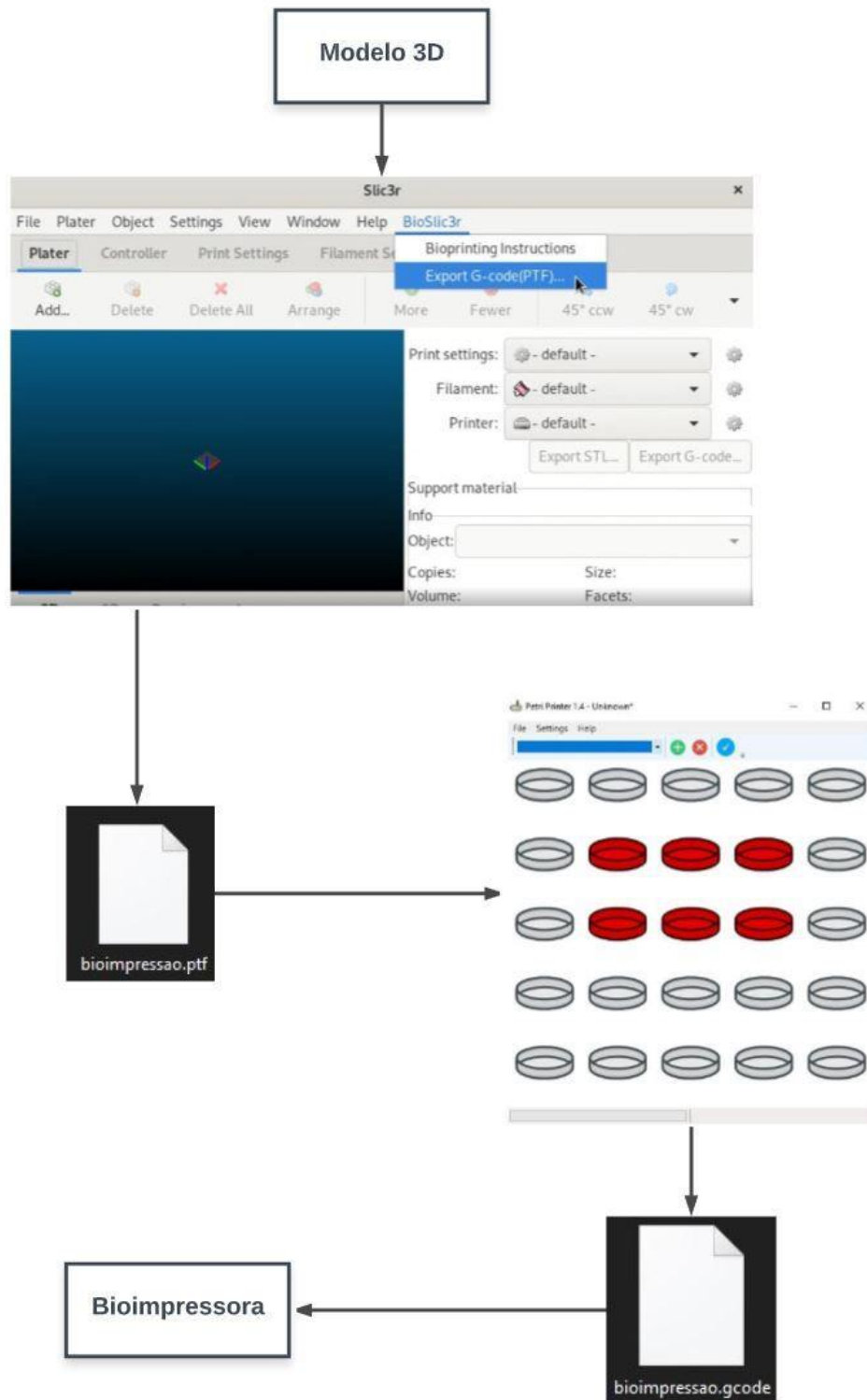


Figura 29 - Novo fluxo de bioimpressão após desenvolvimento da funcionalidade para o Slic3r

6 CONCLUSÃO

6.1 PRINCIPAIS CONTRIBUIÇÕES

Após a análise da pesquisa acadêmica e dados coletados, seguimos para a fase de contribuição, que constitui em como o levantamento bibliográfico foi relevante para o estudo apresentado e para o desenvolvimento da funcionalidade do BioSlic3r - que consiste em uma contribuição para o *software* Slic3r a fim de torná-lo compatível com a bioimpressão.

Além disso, é importante destacar:

- A pesquisa e levantamento da atual situação da área de bioimpressão 3D dado que o campo é relativamente novo, e conseqüentemente, relativamente escasso;
- Apresentação da revisão ~~da~~ *quasi* sistemática de literatura e através dela, ressaltar e correlacionar os conceitos acerca bioimpressão, engenharia de tecidos e impressão 3D.
- A detecção de erros no *build* da ferramenta no sistema Windows, fato que foi documentado no projeto *open source* do Slic3r para possível resolução por futuros desenvolvedores;
- A validação do manual de desenvolvimento do Slic3r no ambiente Linux Fedora;
- A portabilidade e conversão de toda lógica do software gCodeEditor para funcionar em ambiente Linux, visto que foi desenvolvido apenas para Windows, sendo necessária a adaptação;
- Testes de diferentes estratégias e estruturas dentro de um projeto de software para bioimpressão 3D.
- O desenvolvimento do algoritmo em fase experimental porém funcional, possibilitando a integração da ferramenta PetriPrinter e o Slic3r, com o objetivo de chegar em um fatiador adequado para bioimpressão sem necessidade de uso de softwares terceiros.

6.2 VISÃO GERAL

O presente estudo objetivou e otimizou o fluxo de fatiamento da bioimpressão 3D, reduzindo o número de etapas, o tempo gasto e a dificuldade no processo entre a modelagem e o procedimento final de bioimpressão do material, introduzindo uma melhoria no software de fatiamento *open source* Slic3r.

Foi apresentada uma coletânea relativa à engenharia de tecidos de uma forma geral, que introduz e sumariza bem seus principais conceitos e tópicos adjacentes.

O *software* com a melhoria, no estado atual de desenvolvimento apresenta uma contribuição prática e funcional para a área de bioimpressão 3D, com alto potencial de desenvolvimento.

6.3 PRINCIPAIS DIFICULDADES ENCONTRADAS

No que se refere às dificuldades encontradas, é importante destacar problemas que marcaram:

- A enorme escassez de fontes de conhecimento de *softwares* para bioimpressão 3D. Grande parte dos trabalhos somente citam esses softwares como trabalhos futuros;
- Fraca variedade de artigos científicos na temática de bioimpressão 3D.
- O fato dos *softwares* PetriPrinter e gCodeEditor funcionarem apenas em ambiente Windows, pois o desenvolvimento foi feito por completo em Linux por ser um sistema mais propício para programação e com isso foi necessário adaptar todo o ambiente, uma tarefa de extrema complexidade;
- A grande dificuldade de se conseguir configurar o ambiente no sistema operacional Windows. Muitos *links* estavam inoperantes, programas em versões antigas e não funcionais e tutoriais com códigos defasados;

6.4 TRABALHOS FUTUROS

Esse trabalho pode ser considerado um pontapé inicial e uma boa base para o começo de algumas outras abordagens dentro da área de bioimpressão. É sabido que os estudos sobre bioimpressão são relativamente recentes e que esses estudos têm aumentado dado à sua

demanda na medicina. A otimização proposta neste trabalho ainda pode ser muito melhorada, de forma que uma integração ainda mais completa possa ser realizada. Outros algoritmos podem ser criados a partir do trabalho existente também, já que novas demandas surgem conforme as pesquisas avançam. De modo geral, toda dificuldade encontrada e relatada nos capítulos e seções anteriores podem ser temas de trabalhos futuros. Um bom exemplo é o suporte da ferramenta no sistema operacional Windows, que encontra problemas desde a fase de *build*, até a utilização de ferramentas adequadas para desenvolvimento, conforme citado.

A ideia principal é que no futuro, tenhamos um fatiador com suporte nativo à bioimpressão 3D. Com um maior desenvolvimento da ferramenta apresentada nesta pesquisa, acreditamos que isso seja possível de acontecer.

REFERÊNCIAS

- ARSLAN-YILDIZ, A. et al. Towards artificial tissue models: past, present, and future of 3D bioprinting. **iopscience**, 2016. Disponível em: <<https://iopscience.iop.org/article/10.1088/1758-5090/8/1/014103>>. Acesso em: 23/11/2019.
- MIRONOV, V. et al. **3D bioprinting of the kidney—hype or hope?**. Disponível em: <<https://www.aimspress.com/article/10.3934/celltissue.2018.3.119>>. Acesso em: 23/11/2019.
- HÖLZL, K. et al. Bioink properties before, during and after 3D bioprinting. **iopscience**, 2016. Disponível em: <<https://iopscience.iop.org/article/10.1088/1758-5090/8/1/014103>>. Acesso em: 23/11/2019.
- DERNOWSEK, J. A.; REZENDE, R. A.; SILVA, J. V. L., **inctregenera**, 2017. Disponível em: <<http://www.inctregenera.org.br/tecnologia-da-informaccedilatildeo.html>>. Acesso em: 23/11/2019.
- IRVINE, S. A. Bioprinting and Differentiation of Stem Cells. **mdpi**, 2016. Disponível em: <<https://www.mdpi.com/1420-3049/21/9/1188/htm>>. Acesso em: 23/11/2019.
- HINTON, T. J. et al. Three-dimensional printing of complex biological structures by freeform reversible embedding of suspended hydrogels. **ncbi**, 2015. Disponível em: <<https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4646826/>>. Acesso em: 23/11/2019.
- GUILLEMOT, F.; MIRONOV, V.; NAKAMURA, M. Bioprinting is coming of age: report from the International Conference on Bioprinting and Biofabrication in Bordeaux (3B'09). **iopscience**, 2016. Disponível em: <<https://iopscience.iop.org/article/10.1088/1758-5082/2/1/010201>>. Acesso em: 23/11/2019.

SANJAIRAJ, V. et al. 3D Printing and 3D Bioprinting in Pediatrics. **researchgate**, 2017.

Disponível em:

<https://www.researchgate.net/publication/318815696_3D_Printing_and_3D_Bioprinting_in_Pediatrics>. Acesso em: 23/11/2019.

DERNOWSEK, J. **biofabricacao**, 2019. Disponível em:

<<https://www.biofabricacao.com/bioimpressao-3d>>. Acesso em: 27/11/2019.

DERNOWSEK, J. **biofabricacao**, 2019. Disponível em:

<<https://www.biofabricacao.com/single-post/2018/11/04/O-que-s%C3%A3o-Bioinks-biotintas-Conceitos-importantes-da-Bioimpress%C3%A3o-de-tecidos>>. Acesso em: 27/11/2019.

DERNOWSEK, J.; REZENDE, R. A.; SILVA, J. V. L. **futuremedicine**, 2017. Disponível em: <The role of information technology in the future of 3D biofabrication>. Acesso em: 03/12/2019.

COSTA, A. B.; ZOLTOWSKI A. P. C. **researchgate**, 2014. Disponível em:<https://www.researchgate.net/publication/323255862_Como_escrever_um_artigo_de_revisao_sistematica>. Acesso em: 26/06/2019.

GULYAS, M. et al. **Software tools for cell culture-related 3D printed structures**.

Disponível em:<<https://doi.org/10.1371/journal.pone.0203203>>. Acessado em: 25/11/2019

SREEHITHA, V. “Impact of 3D printing in automotive industry” **International Journal of Mechanical and Production Engineering**, Vol.5, No.2, pp. 91-94, 2017.

PETCH, M. et al. Audi gives update on use of SLM metal 3D printing for the automotive industry. **3D Printing Industry**, 2018.

Hubs. 3D et al Design Rules For 3D Printing. **core77**, 2018. Disponível em:

<<https://www.core77.com/posts/74401/Design-Rules-for-3D-Printing>> Acesso em: 01/12/2019.

O'BRIEN, F. J. et al. Biomaterials & scaffolds for tissue engineering. **sciencedirect**, 2011. Disponível em: <<https://www.sciencedirect.com/science/article/pii/S136970211170058X>> Acesso em: 22/11/2019

FERMEIRO, J. B. L. “State of the art and challenges in bioprinting technologies, contribution of the 3D bioprinting in Tissue Engineering”. **ieee**, 2015. Disponível em: <<https://ieeexplore.ieee.org/document/7088883/authors#authors>> Acesso em: 15/10/2019

FURTH, M. E. “Tissue Engineering: Future Perspectives”. **sciencedirect**, 2014. Disponível em: <<https://www.sciencedirect.com/science/article/pii/B9780123983589000069>> Acesso em: 12/11/2019

GIBSON, I. “Additive manufacturing technologies: 3D printing, rapid prototyping, and direct digital manufacturing, second edition”. **researchgate**, 2014. Disponível em: <https://www.researchgate.net/publication/283769646_Additive_manufacturing_technologies_3D_printing_rapid_prototyping_and_direct_digital_manufacturing_second_edition> Acesso em: 01/11/2019

SHAHRUBUDIN, N. “An Overview on 3D Printing Technology: Technological, Materials, and Applications”. **sciencedirect**, 2019. Disponível em: <<https://www.sciencedirect.com/science/article/pii/S2351978919308169>> Acesso em: 10/09/2019

DUDA, T. “3D Metal Printing Technology”. **sciencedirect**, 2016. Disponível em: <<https://www.sciencedirect.com/science/article/pii/S2405896316325496>> Acesso em: 18/10/2019

ROBU, N. “New software tools for hydrogel-based bioprinting”, **IEEE 12th International Symposium on Applied Computational Intelligence and Informatics**, 2019

APÊNDICES

APÊNDICE A – FUNÇÕES PRINCIPAIS

MainFrame.pm

```

...
# BioSlic3r menu
my $bioSlic3r = Wx::Menu->new;
{
    wxTheApp->append_menu_item($bioSlic3r, "&Bioprinting Instructions", 'Show
bioprinting recommendations', sub {
        wxTheApp->bp;
    });
    $bioSlic3r->AppendSeparator();
    wxTheApp->append_menu_item($bioSlic3r, "Export PTF File", 'Export current plate
in PTF format.', sub {
        $self->{plater}->export_ptf;
    }, undef, 'cog_go.png');
}

# menubar
# assign menubar to frame after appending items, otherwise special items
# will not be handled correctly
{
    my $menubar = Wx::MenuBar->new;
    $menubar->Append($fileMenu, "&File");
    $menubar->Append($self->{plater_menu}, "&Plater") if $self->{plater_menu};
    $menubar->Append($self->{object_menu}, "&Object") if $self->{object_menu};
    $menubar->Append($settingsMenu, "&Settings");
    $menubar->Append($self->{viewMenu}, "&View") if $self->{viewMenu};
    $menubar->Append($windowMenu, "&Window");
    $menubar->Append($helpMenu, "&Help");
    $menubar->Append($bioSlic3r, "&BioSlic3r");
    $self->SetMenuBar($menubar);
}
...

```

Plater.pm

```

...
#copying function below of "export_gcode" and adapting for ptf file
sub export_ptf {
  my ($self, $output_file) = @_ ;

  return if !@{$self->{objects}} ;

  if ($self->{export_gcode_output_file}) {
    Wx::MessageDialog->new($self, "Another export job is currently running.", 'Error',
wxOK | wxICON_ERROR)->ShowModal;
    return;
  }

  # if process is not running, validate config
  # (we assume that if it is running, config is valid)
  eval {
    # this will throw errors if config is not valid
    $self->config->validate;
    $self->{print}->validate;
  };
  Slic3r::GUI::catch_error($self) and return;

  # apply config and validate print
  my $config = $self->config;
  eval {
    # this will throw errors if config is not valid
    $config->validate;
    $self->{print}->apply_config($config);
    $self->{print}->validate;
  };
  if (!$Slic3r::have_threads) {
    Slic3r::GUI::catch_error($self) and return;
  }

  # select output file
  if ($output_file) {

```



```

    $self->{export_gcode_output_file} = $self->{print}->output_filepath($output_file);
} else {
    my $default_output_file = $self->{print}->output_filepath($main::opt{output} // "");
    my $dlg = Wx::FileDialog->new($self, 'Save G-code file as:',
wxTheApp->output_path(dirname($default_output_file)),
    basename($default_output_file), &Slic3r::GUI::FILE_WILDCARDS->{gcode},
wxFD_SAVE | wxFD_OVERWRITE_PROMPT);
    if ($dlg->ShowModal != wxID_OK) {
        $dlg->Destroy;
        return;
    }
    my $path = Slic3r::decode_path($dlg->GetPath);
    $Slic3r::GUI::Settings->{_} {last_output_path} = dirname($path);
    wxTheApp->save_settings;
    $self->{export_gcode_output_file} = $path;
    $dlg->Destroy;
}

$self->statusbar->StartBusy;

if ($Slic3r::have_threads) {
    $self->statusbar->SetCancelCallback(sub {
        $self->stop_background_process;
        $self->statusbar->SetStatusText("Export cancelled");
        $self->{export_gcode_output_file} = undef;
        $self->{send_gcode_file} = undef;

        # this updates buttons status
        $self->object_list_changed;
    });

    # start background process, whose completion event handler
    # will detect $self->{export_gcode_output_file} and proceed with export
    $self->start_background_process;
} else {
    eval {
        $self->{print}->process;
        $self->{print}->export_ptf(output_file => $self->{export_gcode_output_file});
    };
    my $result = !Slic3r::GUI::catch_error($self);
    $self->on_export_completed($result);
}

```

```

}

# this updates buttons status
$self->object_list_changed;
$self->toggle_print_stats(1);

#Boost::Serialize();
return $self->{export_gcode_output_file};
}

```

...

BioPrintingDialog.cpp

```

#include "Dialogs/BioPrintingDialog.hpp"

namespace Slic3r { namespace GUI {

static void link_clicked(wxHtmlLinkEvent& e)
{
    wxLaunchDefaultBrowser(e.GetLinkInfo().GetHref());
    e.Skip(0);
}

AboutDialog::AboutDialog(wxWindow* parent) : wxDialog(parent, -1, _("About Slic3rs"),
wxDefaultPosition, wxSize(600, 460), wxCAPTION)
{
    auto* hsizer {new wxBoxSizer(wxHORIZONTAL)} ;

    auto* vsizer {new wxBoxSizer(wxVERTICAL)} ;

    // logo
    auto* logo {new AboutDialogLogo(this)};
    hsizer->Add(logo, 0, wxEXPAND | wxLEFT | wxRIGHT, 30);

    // title
    auto* title { new wxStaticText(this, -1, "Slic3r", wxDefaultPosition,
wxDefaultSize)};
    auto title_font { wxSystemSettings::GetFont(wxSYS_DEFAULT_GUI_FONT)};

    title_font.SetWeight(wxFONTWEIGHT_BOLD);

```

```

title_font.SetFamily(wxFONTFAMILY_ROMAN);
title_font.SetPointSize(24);
title->SetFont(title_font);

vsizer->Add(title, 0, wxALIGN_LEFT | wxTOP, 30);

// version

auto* version {new wxStaticText(this, -1, wxString("Version ") +
wxString(SLIC3R_VERSION), wxDefaultPosition, wxDefaultSize)};
auto version_font { wxSystemSettings::GetFont(wxSYS_DEFAULT_GUI_FONT) };
version_font.SetPointSize((the_os == OS::Windows ? 9 : 11));
version->SetFont(version_font);
vsizer->Add(version, 0, wxALIGN_LEFT | wxBOTTOM, 10);

// text

wxString text {""};
text << "<html>"
  << "<body>"
  << "Copyright &copy; 2011-2017 Alessandro Ranellucci. Bernardo !!!!! <br />"
  << "<a href='\"http://slic3r.org^\">Slic3r</a> is licensed under the "
  << "<a href='\"http://www.gnu.org/licenses/agpl-3.0.html\">GNU Affero General
Public License, version 3</a>."
  << "<br /><br /><br />"
  << "Contributions by Henrik Brix Andersen, Vojtech Bubnik, Nicolas Dandrimont,
Mark Hindess, "
  << "Petr Ledvina, Joseph Lenox, Y. Sapir, Mike Sheldrake, Kliment Yanev and
numerous others. "
  << "Manual by Gary Hodgson. Inspired by the RepRap community. <br />"
  << "Slic3r logo designed by Corey Daniels, <a
href='\"http://www.famfamfam.com/lab/icons/silk^\">Silk Icon Set</a> designed by Mark
James. "
  << "<br /><br />"
  << "Built on " << build_date << " at git version " << git_version << "."
  << "</body>"
  << "</html>";

auto* html {new wxHtmlWindow(this, -1, wxDefaultPosition, wxDefaultSize,
wxHW_SCROLLBAR_NEVER)};
html->SetBorders(2);
html->SetPage(text);

```

```

    html->Bind(wxEVT_HTML_LINK_CLICKED, [=](wxHtmlLinkEvent& e){
link_clicked(e); });

```

```

    vsizer->Add(html, 1, wxEXPAND | wxALIGN_LEFT | wxRIGHT | wxBOTTOM,
20);

```

```

    // buttons

```

```

    auto buttons = this->CreateStdDialogButtonSizer(wxOK);

```

```

    this->SetEscapeId(wxID_CLOSE);

```

```

    vsizer->Add(buttons, 0, wxEXPAND | wxRIGHT | wxBOTTOM, 3);

```

```

    hsizer->Add(vsizer, 1, wxEXPAND, 0);

```

```

    this->SetSizer(hsizer);

```

```

};

```

```

AboutDialogLogo::AboutDialogLogo(wxWindow* parent) :

```

```

    wxPanel(parent, -1, wxDefaultPosition, wxDefaultSize)

```

```

{

```

```

    this->logo = wxBitmap(var("Slic3r_192px.png"), wxBITMAP_TYPE_PNG);

```

```

    this->SetMinSize(wxSize(this->logo.GetWidth(), this->logo.GetHeight()));

```

```

    this->Bind(wxEVT_PAINT, [=](wxPaintEvent& e) { this->repaint(e);});

```

```

}

```

```

void AboutDialogLogo::repaint(wxPaintEvent& event)

```

```

{

```

```

    wxPaintDC dc {this};

```

```

    dc.SetBackgroundMode(wxPENSTYLE_TRANSPARENT);

```

```

    const wxSize size {this->GetSize()} ;

```

```

    const auto logo_w {this->logo.GetWidth()};

```

```

    const auto logo_h {this->logo.GetHeight()};

```

```

    dc.DrawBitmap(this->logo, (size.GetWidth() - logo_w) / 2, (size.GetHeight() - logo_h) /
2, 1);

```

```

    event.Skip();

```

```

}

```

```
}} // namespace Slic3r::GUI
```

```
...
```

BioPrintingDialog.hpp

```
#ifndef ABOUTDIALOG_HPP
#define ABOUTDIALOG_HPP
#include <wx/dialog.h>
#include <wx/sizer.h>
#include <wx/settings.h>
#include <wx/colour.h>
#include <wx/html/htmlwin.h>
#include <wx/event.h>
#include <string>

#include "libslic3r.h"
#include "misc_ui.hpp"

#ifndef SLIC3R_BUILD_COMMIT
#define SLIC3R_BUILD_COMMIT (Unknown revision)
#endif

#define VER1_(x) #x
#define VER_(x) VER1_(x)
#define BUILD_COMMIT VER_(SLIC3R_BUILD_COMMIT)

namespace Slic3r { namespace GUI {

const wxString build_date {__DATE__};
const wxString git_version {BUILD_COMMIT};

class AboutDialogLogo : public wxPanel {
private:
    wxBitmap logo;
public:
    AboutDialogLogo(wxWindow* parent);
    void repaint(wxPaintEvent& event);
};

class BioPrintingDialog : public wxDialog {
```

```
public:
    // Build and show the About popup.
    BioPrintingDialog(wxWindow* parent);
};
```

```
}} // namespace Slic3r::GUI
```

```
#endif // ABOUTDIALOG_HPP
```

```
...
```

GUI.pm

```
package Slic3r::GUI;
use strict;
use warnings;
use utf8;

use Wx 0.9901 qw(:bitmap :dialog :icon :id :misc :systemsettings :toplevelwindow
    :filedialog :font);
use Wx::Event qw(EVT_MENU);

BEGIN {
    # Wrap the Wx::_load_plugin() function which doesn't work with non-ASCII paths
    no warnings 'redefine';
    my $orig = *Wx::_load_plugin{CODE};
    *Wx::_load_plugin = sub {
        $_[0] = Slic3r::decode_path($_[0]);
        $orig->(@_);
    };
}

use File::Basename qw(basename);
use FindBin;
use List::Util qw(first any);
use Slic3r::Geometry qw(X Y);

use Slic3r::GUI::2DBed;
use Slic3r::GUI::AboutDialog;
```

```

use Slic3r::GUI::BedShapeDialog;
use Slic3r::GUI::BioPrintingDialog;
use Slic3r::GUI::BonjourBrowser;
use Slic3r::GUI::ConfigWizard;
use Slic3r::GUI::Controller;
use Slic3r::GUI::Controller::ManualControlDialog;
use Slic3r::GUI::Controller::PrinterPanel;
use Slic3r::GUI::MainFrame;
use Slic3r::GUI::Notifier;
use Slic3r::GUI::Plater;
use Slic3r::GUI::Plater::2D;
use Slic3r::GUI::Plater::2DToolpaths;
use Slic3r::GUI::Plater::3D;
use Slic3r::GUI::Plater::3DPreview;
use Slic3r::GUI::Plater::ObjectPartsPanel;
use Slic3r::GUI::Plater::ObjectCutDialog;
use Slic3r::GUI::Plater::ObjectRotateFaceDialog;
use Slic3r::GUI::Plater::ObjectSettingsDialog;
use Slic3r::GUI::Plater::LambdaObjectDialog;
use Slic3r::GUI::Plater::OverrideSettingsPanel;
use Slic3r::GUI::Plater::SplineControl;
use Slic3r::GUI::Preferences;
use Slic3r::GUI::ProgressStatusBar;
use Slic3r::GUI::Projector;
use Slic3r::GUI::OptionsGroup;
use Slic3r::GUI::OptionsGroup::Field;
use Slic3r::GUI::Preset;
use Slic3r::GUI::PresetEditor;
use Slic3r::GUI::PresetEditorDialog;
use Slic3r::GUI::SLAPrintOptions;
use Slic3r::GUI::ReloadDialog;

our $have_OpenGL = eval "use Slic3r::GUI::3DScene; 1";
our $have_LWP = eval "use LWP::UserAgent; 1";

use Wx::Event qw(EVT_IDLE EVT_COMMAND);
use base 'Wx::App';

use constant FILE_WILDCARDS => {
    known => 'Known files (*.stl, *.obj, *.amf, *.xml,
*.3mf)|*.3mf;*.3MF;*.stl;*.STL;*.obj;*.OBJ;*.amf;*.AMF;*.xml;*.XML',

```

```

stl    => 'STL files (*.stl)|*.stl;*.STL',
obj    => 'OBJ files (*.obj)|*.obj;*.OBJ',
amf    => 'AMF files (*.amf)|*.amf;*.AMF;*.xml;*.XML',
tmf    => '3MF files (*.3mf)|*.3mf;*.3MF',
ini    => 'INI files *.ini|*.ini;*.INI',
gcode  => 'G-code files (*.gcode, *.gco, *.g,
*.ngc)|*.gcode;*.GCODE;*.gco;*.GCO;*.g;*.G;*.ngc;*.NGC',
svg    => 'SVG files *.svg|*.svg;*.SVG',
};

use constant MODEL_WILDCARD => join '|', @{{&FILE_WILDCARDS}} {qw(known stl
obj amf tmf)};
use constant STL_MODEL_WILDCARD => join '|', @{{&FILE_WILDCARDS}} {qw(stl)};
use constant AMF_MODEL_WILDCARD => join '|',
@{{&FILE_WILDCARDS}} {qw(amf)};
use constant TMF_MODEL_WILDCARD => join '|',
@{{&FILE_WILDCARDS}} {qw(tmf)};

our $datadir;
# If set, the "Controller" tab for the control of the printer over serial line and the serial port
settings are hidden.
our $autosave;
our $threads;
our @cb;

our $Settings = {
    _ => {
        version_check => 1,
        autocenter => 1,
        autoalignz => 1,
        invert_zoom => 0,
        background_processing => 0,
        threads => $Slic3r::Config::Options->{threads} {default},
        color_toolpaths_by => 'role',
        tabbed_preset_editors => 1,
        show_host => 1,
        nudge_val => 1,
        rotation_controls => 'z',
        reload_hide_dialog => 0,
        reload_behavior => 0
    },
};

```



```

our $have_button_icons = &Wx::wxVERSION_STRING =~ / (?:2\.\.9\.[1-9]|3\.) /;
our $small_font = Wx::SystemSettings::GetFont(wxSYS_DEFAULT_GUI_FONT);
$small_font->SetPointSize(11) if &Wx::wxMAC;
our $small_bold_font = Wx::SystemSettings::GetFont(wxSYS_DEFAULT_GUI_FONT);
$small_bold_font->SetPointSize(11) if &Wx::wxMAC;
$small_bold_font->SetWeight(wxFONTWEIGHT_BOLD);
our $medium_font = Wx::SystemSettings::GetFont(wxSYS_DEFAULT_GUI_FONT);
$medium_font->SetPointSize(12);
our $grey = Wx::Colour->new(200,200,200);

# to use in ScrolledWindow::SetScrollRate(xstep, ystep)
# step related to system font point size
our $scroll_step =
Wx::SystemSettings::GetFont(wxSYS_DEFAULT_GUI_FONT)->GetPointSize;

our $VERSION_CHECK_EVENT : shared = Wx::NewEventType;

our $DLP_projection_screen;

sub OnInit {
    my ($self) = @_ ;

    $self->SetAppName('Slic3r');
    Slic3r::debugf "wxWidgets version %s, Wx version %s\n",
    &Wx::wxVERSION_STRING, $Wx::VERSION;

    $self->{notifier} = Slic3r::GUI::Notifier->new;
    $self->{presets} = { print => [], filament => [], printer => [] };

    # locate or create data directory
    # Unix: ~/.Slic3r
    # Windows: "C:\Users\username\AppData\Roaming\Slic3r" or "C:\Documents and
Settings\username\Application Data\Slic3r"
    # Mac: "~/Library/Application Support/Slic3r"
    $datadir ||= Slic3r::decode_path(Wx::StandardPaths::Get->GetUserDataDir);
    my $enc_datadir = Slic3r::encode_path($datadir);
    Slic3r::debugf "Data directory: %s\n", $datadir;

    # just checking for existence of $datadir is not enough: it may be an empty directory
    # supplied as argument to --datadir; in that case we should still run the wizard

```

```

my $run_wizard = (-d $enc_datadir && -e "$enc_datadir/slic3r.ini") ? 0 : 1;
foreach my $dir ($enc_datadir, "$enc_datadir/print", "$enc_datadir/filament",
"$enc_datadir/printer") {
    next if -d $dir;
    if (!mkdir $dir) {
        my $error = "Slic3r was unable to create its data directory at $dir ($!).";
        warn "$error\n";
        fatal_error(undef, $error);
    }
}

# load settings
my $last_version;
if (-f "$enc_datadir/slic3r.ini") {
    my $ini = eval { Slic3r::Config->read_ini("$datadir/slic3r.ini") };
    if ($ini) {
        $last_version = $ini->{ } {version};
        $ini->{ } {$_} = $Settings->{ } {$_}
        for grep !exists $ini->{ } {$_}, keys %{$Settings->{ } };
        $Settings = $ini;
    }
    delete $Settings->{ } {mode}; # handle legacy
}
$Settings->{ } {version} = $Slic3r::VERSION;
$Settings->{ } {threads} = $threads if $threads;
$self->save_settings;

if (-f "$enc_datadir/simple.ini") {
    # The Simple Mode settings were already automatically duplicated to presets
    # named "Simple Mode" in each group, so we already support retrocompatibility.
    unlink "$enc_datadir/simple.ini";
}

$self->load_presets;

# application frame
Wx::Image::AddHandler(Wx::PNGHandler->new);
$self->{mainframe} = my $frame = Slic3r::GUI::MainFrame->new;
$self->SetTopWindow($frame);

# load init bundle

```

```

{
  my @dirs = ($FindBin::Bin);
  if (&Wx::wxMAC) {
    push @dirs, qw();
  } elsif (&Wx::wxMSW) {
    push @dirs, qw();
  }
  my $init_bundle = first { -e $_ } map "$_.init_bundle.ini", @dirs;
  if ($init_bundle) {
    Slic3r::debugf "Loading config bundle from %s\n", $init_bundle;
    $self->{mainframe}->load_configbundle($init_bundle, 1);
    $run_wizard = 0;
  }
}

if (!$run_wizard && (!defined $last_version || $last_version ne $Slic3r::VERSION)) {
  # user was running another Slic3r version on this computer
  if (!defined $last_version || $last_version =~ /^0\./) {
    show_info($self->{mainframe}, "Hello! Support material was improved since the "
      . "last version of Slic3r you used. It is strongly recommended to revert "
      . "your support material settings to the factory defaults and start from "
      . "those. Enjoy and provide feedback!", "Support Material");
  }
  if (!defined $last_version || $last_version =~ /^(?:0|1\.[01])\./) {
    show_info($self->{mainframe}, "Hello! In this version a new Bed Shape option
was "
      . "added. If the bed coordinates in the plater preview screen look wrong, go "
      . "to Print Settings and click the \"Set\" button next to \"Bed Shape\".", "Bed
Shape");
  }
}
$self->{mainframe}->config_wizard if $run_wizard;

$self->check_version
  if $self->have_version_check
    && ($Settings->{__} {version_check} // 1)
    && (!$Settings->{__} {last_version_check} || (time -
$Settings->{__} {last_version_check}) >= 86400);

EVT_IDLE($frame, sub {
  while (my $cb = shift @cb) {

```

```

        $cb->());
    }
});

EVT_COMMAND($self, -1, $VERSION_CHECK_EVENT, sub {
    my ($self, $event) = @_;
    my ($success, $response, $manual_check) = @{$event->GetData};

    if ($success) {
        if ($response =~ /^obsolete ?= ?([a-z0-9.-]+)*\Q$Slic3r::VERSION\E(?:,|$)/) {
            my $res = Wx::MessageDialog->new(undef, "A new version is available. Do you
want to open the Slic3r website now?",
                'Update', wxYES_NO | wxCANCEL | wxYES_DEFAULT |
wxICON_INFORMATION | wxICON_ERROR)->ShowModal;
            Wx::LaunchDefaultBrowser("http://slic3r.org") if $res == wxID_YES;
        } else {
            Slic3r::GUI::show_info(undef, "You're using the latest version. No updates are
available.") if $manual_check;
        }
        $Settings->{$_}{last_version_check} = time();
        $self->save_settings;
    } else {
        Slic3r::GUI::show_error(undef, "Failed to check for updates. Try later.") if
$manual_check;
    }
});

return 1;
}

sub bp {
    my ($self) = @_;

    my $bp = Slic3r::GUI::BioPrintingDialog->new(undef);
    $bp->ShowModal;
    $bp->Destroy;
}

sub about {
    my ($self) = @_;

```

```

    my $about = Slic3r::GUI::AboutDialog->new(undef);
    $about->ShowModal;
    $about->Destroy;
}

# static method accepting a wxWindow object as first parameter
sub catch_error {
    my ($self, $cb, $message_dialog) = @_;
    if (my $err = $@) {
        $cb->() if $cb;
        $message_dialog
            ? $message_dialog->($err, 'Error', wxOK | wxICON_ERROR)
            : Slic3r::GUI::show_error($self, $err);
        return 1;
    }
    return 0;
}

# static method accepting a wxWindow object as first parameter
sub show_error {
    my ($parent, $message) = @_;
    Wx::MessageDialog->new($parent, $message, 'Error', wxOK |
wxICON_ERROR)->ShowModal;
}

# static method accepting a wxWindow object as first parameter
sub show_info {
    my ($parent, $message, $title) = @_;
    Wx::MessageDialog->new($parent, $message, $title || 'Notice', wxOK |
wxICON_INFORMATION)->ShowModal;
}

# static method accepting a wxWindow object as first parameter
sub fatal_error {
    show_error(@_);
    exit 1;
}

# static method accepting a wxWindow object as first parameter
sub warning_catcher {
    my ($self, $message_dialog) = @_;

```

```

return sub {
  my $message = shift;
  return if $message =~ /GLUquadricObjPtr|Attempt to free unreferenced scalar/;
  my @params = ($message, 'Warning', wxOK | wxICON_WARNING);
  $message_dialog
    ? $message_dialog->(@params)
    : Wx::MessageDialog->new($self, @params)->ShowModal;
};
}

sub notify {
  my ($self, $message) = @_;

  my $frame = $self->GetTopWindow;
  # try harder to attract user attention on OS X
  $frame->RequestUserAttention(&Wx::wxMAC ? wxUSER_ATTENTION_ERROR :
wxUSER_ATTENTION_INFO)
    unless ($frame->IsActive);

  $self->{notifier}->notify($message);
}

sub save_settings {
  my ($self) = @_;
  Slic3r::Config->write_ini("$datadir/slic3r.ini", $Settings);
}

sub presets { return $_[0]->{presets}; }

sub load_presets {
  my ($self) = @_;

  for my $group (qw(printer filament print)) {
    my $presets = $self->{presets}{$group};

    # keep external or dirty presets
    @$presets = grep { ($_->external && $_->file_exists) || $_->dirty } @$presets;

    my $dir = "$Slic3r::GUI::datadir/$group";
    opendir my $dh, Slic3r::encode_path($dir)
      or die "Failed to read directory $dir (errno: $!)\n";
  }
}

```

```

foreach my $file (grep ^\.ini$/i, readdir $dh) {
    $file = Slic3r::decode_path($file);
    my $name = basename($file);
    $name =~ s/^\.ini$/i;

    # skip if we already have it
    next if any { $_->name eq $name } @$presets;

    push @$presets, Slic3r::GUI::Preset->new(
        group => $group,
        name   => $name,
        file   => "$dir/$file",
    );
}
closedir $dh;

@$presets = sort { $a->name cmp $b->name } @$presets;

unshift @$presets, Slic3r::GUI::Preset->new(
    group => $group,
    default => 1,
    name   => '- default -',
);
}
}

sub add_external_preset {
    my ($self, $file) = @_ ;

    my $name = basename($file); # keep .ini suffix
    for my $group (qw(printer filament print)) {
        my $presets = $self->{presets} {$group};

        # remove any existing preset with the same name
        @$presets = grep { $_->name ne $name } @$presets;

        push @$presets, Slic3r::GUI::Preset->new(
            group   => $group,
            name     => $name,
            file     => $file,
            external => 1,
        );
    }
}

```

```

    );
}
return $name;
}

sub have_version_check {
    my ($self) = @_;

    # return an explicit 0
    return ($Slic3r::have_threads && $Slic3r::VERSION !~ /-dev$/ && $have_LWP) || 0;
}

sub check_version {
    my ($self, $manual_check) = @_;

    Slic3r::debugf "Checking for updates...\n";

    @_ = ();
    threads->create(sub {
        my $ua = LWP::UserAgent->new;
        $ua->timeout(10);
        my $response = $ua->get('http://slic3r.org/updatecheck');
        Wx::PostEvent($self, Wx::PIThreadEvent->new(-1, $VERSION_CHECK_EVENT,
            threads::shared::shared_clone([ $response->is_success,
            $response->decoded_content, $manual_check ])));

        Slic3r::thread_cleanup();
    })->detach;
}

sub output_path {
    my ($self, $dir) = @_;

    return ($Settings->{__} {last_output_path} && $Settings->{__} {remember_output_path})
        ? $Settings->{__} {last_output_path}
        : $dir;
}

sub open_model {
    my ($self, $window) = @_;

```



```

my $dir = $Slic3r::GUI::Settings->{recent} {skein_directory}
    || $Slic3r::GUI::Settings->{recent} {config_directory}
    || "";

my $dialog = Wx::FileDialog->new($window // $self->GetTopWindow, 'Choose one or
more files (STL/OBJ/AMF/3MF):', $dir, "",
    MODEL_WILDCARD, wxFD_OPEN | wxFD_MULTIPLE |
wxFD_FILE_MUST_EXIST);
if ($dialog->ShowModal != wxID_OK) {
    $dialog->Destroy;
    return;
}
my @input_files = map $Slic3r::decode_path($_), $dialog->GetPaths;
$dialog->Destroy;

return @input_files;
}

sub CallAfter {
    my ($self, $cb) = @_;
    push @cb, $cb;
}

sub scan_serial_ports {
    my ($self) = @_;

    my @ports = ();

    if ($^O eq 'MSWin32') {
        # Windows
        if (eval "use Win32::TieRegistry; 1") {
            my $ts =
Win32::TieRegistry->new("HKEY_LOCAL_MACHINE\\HARDWARE\\DEVICEMAP\\
SERIALCOMM",
                { Access => 'KEY_READ' });
            if ($ts) {
                # when no serial ports are available, the registry key doesn't exist and
                # TieRegistry->new returns undef
                $ts->Tie(\my %reg);
                push @ports, sort values %reg;
            }
        }
    }
}

```

```

    }
} else {
    # UNIX and OS X
    push @ports, glob '/dev/{ttyUSB,ttyACM,tty.,cu.,rfcomm} *';
}

return grep !/Bluetooth|FireFly/, @ports;
}

sub append_menu_item {
    my ($self, $menu, $string, $description, $cb, $id, $icon, $kind) = @_;

    $id //= &Wx::NewId();
    my $item = Wx::MenuItem->new($menu, $id, $string, $description // "", $kind // 0);
    $self->set_menu_item_icon($item, $icon);
    $menu->Append($item);

    EVT_MENU($self, $id, $cb);
    return $item;
}

sub append_submenu {
    my ($self, $menu, $string, $description, $submenu, $id, $icon) = @_;

    $id //= &Wx::NewId();
    my $item = Wx::MenuItem->new($menu, $id, $string, $description // "");
    $self->set_menu_item_icon($item, $icon);
    $item->SetSubMenu($submenu);
    $menu->Append($item);

    return $item;
}

sub set_menu_item_icon {
    my ($self, $menuItem, $icon) = @_;

    # SetBitmap was not available on OS X before Wx 0.9927
    if ($icon && $menuItem->can('SetBitmap')) {
        $menuItem->SetBitmap(Wx::Bitmap->new($Slic3r::var->($icon),
wxBITMAP_TYPE_PNG));
    }
}

```

```

}

sub save_window_pos {
    my ($self, $window, $name) = @_;

    $Settings->{$_} {"${name}_pos"} = join '!', $window->GetScreenPositionXY;
    $Settings->{$_} {"${name}_size"} = join '!', $window->GetSizeWH;
    $Settings->{$_} {"${name}_maximized"} = $window->IsMaximized;
    $self->save_settings;
}

sub restore_window_pos {
    my ($self, $window, $name) = @_;

    if (defined $Settings->{$_} {"${name}_pos"}) {
        my $size = [ split '!', $Settings->{$_} {"${name}_size"}, 2 ];
        $window->SetSize($size);

        my $display = Wx::Display->new->GetClientArea();
        my $pos = [ split '!', $Settings->{$_} {"${name}_pos"}, 2 ];
        if (($pos->[X] + $size->[X]/2) < $display->GetRight && ($pos->[Y] + $size->[Y]/2)
        < $display->GetBottom) {
            $window->Move($pos);
        }
        $window->Maximize(1) if $Settings->{$_} {"${name}_maximized"};
    }
}

1;

```