TRABALHO DE GRADUAÇÃO

PROJETO DE UM SIMULADOR DE MÁQUINAS CNC EM PLATAFORMA WEB

Filipe Alves Caixeta

Brasília, novembro de 2016

ENGENHARIA
MECATRÔNICA
UNIVERSIDADE DE BRASÍLIA

UNIVERSIDADE DE BRASILIA
Faculdade de Tecnologia
Curso de Graduação em Engenharia de Controle e Automação

TRABALHO DE GRADUAÇÃO

PROJETO DE UM SIMULADOR DE MÁQUINAS CNC EM PLATAFORMA WEB

Filipe Alves Caixeta

Relatório submetido como requisito parcial de obtenção de grau de Engenheiro de Controle e Automação

Banca Examinadora

Professor Alexandre Zaghetto, CIC/UnB _____
Orientador

Professora Aida Alves Fadel, ENM/UnB _____
Examinador interno

Mestre Miguel Eduardo Gutierrez, ENM/UnB _____
Examinador interno

Brasília, novembro de 2016

FICHA CATALOGRÁFICA

CAIXETA, FILIPE ALVES.
Projeto de um simulador de máquinas CNC em plataforma web,

[Distrito Federal] 2016.

x, 53p., 297 mm (FT/UnB, Engenheiro, Controle e Automação, 2016). Trabalho de Graduação – Universidade de Brasília.Faculdade de Tecnologia.

1. Simulador CNC 2.Torno CNC
3. Fresadora CNC 4.Impressora 3D

I. Mecatrônica/FT/UnB II. Projeto de um simulador de máquinas CNC em plataforma web

REFERÊNCIA BIBLIOGRÁFICA

CAIXETA, F. A., (2016). Projeto de um simulador de máquinas CNC em plataforma web. Trabalho de Graduação em Engenharia de Controle e Automação, Publicação FT.TG-$n°$017/2016, Faculdade de Tecnologia, Universidade de Brasília, Brasília, DF, 53p.

CESSÃO DE DIREITOS

AUTOR: Filipe Alves Caixeta

TÍTULO DO TRABALHO DE GRADUAÇÃO: Projeto de um simulador de máquinas CNC em plataforma web.

GRAU: Engenheiro ANO: 2016

Filipe Alves Caixeta

Brasília – DF – Brasil.

Dedicatória

Dedico este trabalho a minha familia e aos meus amigos.

Filipe Alves Caixeta

Agradecimentos

Agradeço a todos os meus colegas do curso de engenharia mecatrônica, principalmente a turma 27 que sempre esteve unida nos momentos de procrastinação e nos momentos de virar a noite estudando. As amizades que fiz durante esse tempo fizeram com que o tempo passasse de forma agradável e divertida.

Agradeço a minha família que sempre me deu suporte para continuar e chegar onde cheguei hoje. Me ensinaram a não andar pelas circunstancias e não desistir dos meus sonhos.

Agradeço a todos os amigos que colaboraram com meu TG de várias formas. Jesse, Andrezinho, Primo, Rafael, Boson, Victor Matheus, Caio, Rodrigo, Cris, Guilherme e Redy foram alguns dos que ajudaram dando sugestões, utilizando o simulador ou contribuindo com imagens e códigos de exemplo utilizados neste trabalho.

Agradeço a Geordana que me deu carona várias vezes até o CIC, me ajudou com a revisão do TG e também pela companhia.

Agradeço também ao meu orientador Professor Dr. Alexandre Zaghetto que acreditou e viu potencial no meu projeto desde o início.

Filipe Alves Caixeta

RESUMO

Com a grande popularização das impressoras 3D e outras máquinas de prototipagem rápidas, estamos experimentando uma nova revolução industrial, na qual tudo pode ser prototipado ou fabricado em casa. Máquinas como impressoras 3D, tornos e fresadoras trabalham com programação em código-G para automatização das tarefas. Este trabalho aborda o projeto e implementação de um simulador de máquinas de comando numérico através da programação em código-G norma RS274D. O simulador, CNC Web Simulator, foi implementado utilizando tecnologias web para que fosse possível acessar em qualquer sistema operacional, incluindo dispositivos moveis, ultrapassando as limitações de outros softwares exclusivos para plataforma Microsoft Windows. O CNC Web Simulator apresentou resultados muito bons, com ótimas renderizaçoes 3D e velocidade de simulação maior que muitos simuladores proprietários.

Palavras Chave: CNC, Impressora 3D, Torno, Fresadora.

ABSTRACT

3D printers and prototyping machines have become more popular now, some speak of a new industrial revolution where anything will be prototyped or manufactured at home. Machines like 3D printers, lathe and milling are programed in a language called G-code. The propose of this project is to implement a simulator for computer numeric control machines programed by G-code oder RS274D.The simulator, named CNC Web Simulator, was implemented using web technologies to make it possible to run on any operating system, including mobile devices, overcoming the limitations of proprietary softwares that are exclusively for Microsoft Windows platforms. CNC Web Simulator showed good results, with great 3D rendering images, and simulation speed faster than many proprietary simulators.

Keywords: CNC, 3D Printer, Lathe, Milling.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SÍMBOLOS

Siglas

API	Interface de Programação de Aplicações - *Application Programming Interface*
CAD	Desenho assistido por computador - *Computer Aided Design*
CAM	Manufatura assistida por computador - *Computer Aided Manufacturing*
CN	Comando numérico - *Numeric Control*
CNC	Comando numérico computadorizado - *Computer Numeric Control*
CPU	Unidade de processamento central - *Central Processing Unit*
GPU	Unidade de Processamento Gráfico - *Graphics Processing Unit*

Capítulo 1

Introdução

1.1 Contextualização

O torno mecânico é uma máquina que trabalha com uma ferramenta com aresta cortante sobre uma peça em rotação. Existem relatos de tornos para fabricação de cerâmica datados de mais de 3000 anos. Durante este período, o torno passou por várias modificações e até mesmo Leonardo da Vinci fez sua contribuição em um projeto de um torno que poderia ser operado por uma única pessoa, utilizando um sistema parecido com o das máquinas de costura manual. Com o surgimento do motor elétrico, os tornos foram aperfeiçoados mais uma vez [5]. A Figura 1.1 mostra um torno mecânico com operação manual.

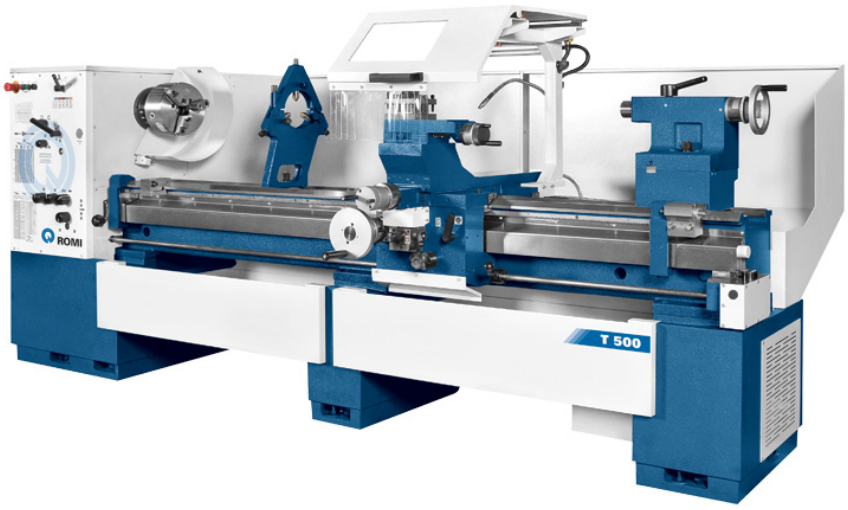


Figura 1.1: Torno mecanico universal linha ROMI T. [6]

Em 1818, com a necessidade de fabricação de uma grande quantidade de peças de rifles, o norte americano Eli Whitney criou a fresadora [7]. A fresadora é uma máquina de usinagem que possui movimentação cartesiana em três eixos e que faz a usinagem do material utilizando uma ferramenta de aresta de corte relativa. Ao longo do tempo, a fresadora foi se modificando até chegar nas fresadoras atuais como pode ser visto na Figura 1.2

Figura 1.2: Fresadora vertical Central Machinery.[8]

Na década de 50, com o surgimento dos computadores, o torno e a fresadora passaram a permitir procedimentos automatizados através da programação dos seus movimentos axiais. Inicialmente, as máquinas trabalhavam com cartões perfurados. As máquinas que trabalham com os comandos computadorizados passaram a se chamar máquinas CNC (Comando Numérico Computadorizado). Atualmente, a tecnologia tem se desenvolvido ao ponto de existirem programas que transformam um desenho em comandos de usinagem para tornos e fresadoras CNC programáveis. Estes programas são chamados CAM (Manufatura Assistida por Computador). A linguagem de comandos mais conhecida e adotada para as máquinas CNC é chamada de código-G com revisão final aprovada em fevereiro de 1980 como RS274D [9].

Além do torno e da fresadora, outras máquinas também passaram a utilizar o comando numérico computadorizado. Algumas delas são: a impressora 3D, a cortadora a laser, as máquinas para dobrar chapas ou tubos, as máquinas para usinar placas de circuito elétrico, etc. Apesar das impressoras 3D terem sido inventadas nos anos 80 como tecnologia de prototipagem rápida, somente em 2009, as primeiras impressoras começaram a ser comercializadas. Atualmente, as impressoras 3D têm se popularizado rapidamente. Assim como a fresadora, as impressoras 3D trabalham de maneira muito similar e também com código-G. Por ser algo muito recente, cada fabricante tem adotado variações diferentes do código-G [10]. Um exemplo de impressora 3D pode ser visto na Figura1.3

Figura 1.3: Impressora 3D MakerBot Replicator.[1]

1.2 Definição do problema

Atualmente, o *software* CNC Simulator é uma das poucas formas viáveis para alunos estudarem a programação de máquinas CNC. A maioria dos simuladores de máquinas CNC são *softwares* CAD/CAM ou CAM e já geram o código-G automaticamente através de uma série de especificações definidas pelo usuário. Para um aluno que deseja aprender a trabalhar com máquinas CNC, é importante trabalhar inicialmente fora de um *software* CAM para entender a sintaxe e a estrutura da linguagem de código-G para, em seguida, entender como um *software* CAM funciona. Infelizmente, a maior parte dos simuladores são para a plataforma Microsoft Windows, o que obriga usuários de outros sistemas operacionais a utilizarem máquinas virtuais ou instalarem o Windows para realizarem simulações de operações em máquinas CNC. Neste trabalho, será proposto um simulador web didático de máquinas CNC multiplataforma chamado de CNC Web Simulator.

1.3 Objetivos do projeto

O objetivo deste trabalho é projetar e implementar um simulador CNC que possa ser utilizado para ensino de código-G e máquinas CNC em uma plataforma que funcione não somente em ambiente Windows, mas em qualquer sistema operacional, incluindo *tablets* e *smartphones*. Por ser inviável implementar um simulador para todos os tipos de máquinas CNC, este trabalho só implementa a simulação de tornos, fresadoras e impressoras 3D. É também importante que o simulador seja didático e de fácil uso para estudantes.

1.4 Descrição do documento

Este trabalho é dividido em cinco capítulos, sendo o primeiro deles esta Introdução. O Capítulo 2 apresenta a fundamentação teórica do trabalho, passando pela teoria da computação gráfica e do funcionamento tanto de máquinas, quanto de simuladores CNC. O Capítulo 3 apresenta a interface gráfica do simulador criado, o resumo do funcionamento e a organização do código. No Capítulo 4, é possível ver algumas imagens feitas no simulador implementado, comparações com outros simuladores e uma análise dos resultados. Por fim, o Capítulo 5 apresenta as conclusões acerca do trabalho e sugestões para trabalhos futuros. Em anexo, há um manual de usuário explicando o necessário para utilizar o simulador.

Capítulo 2

Fundamentação teórica

2.1 Computação gráfica

A computação gráfica aborda todos os aspectos da criação e da manipulação de imagens. As principais áreas da computação gráfica são:

1. Visualização de informação
2. Design
3. Simulação e animação
4. Interfaces de usuário

A área da computação gráfica é uma área multidisciplinar que envolve áreas como física, engenharia, design, computação, artes e muitas outras. A computação gráfica tem grande importância na área médica, na qual dados provenientes de ultrassom, tomografia, ressonância magnética e outros são transformados em imagens para facilitar a visualização dos dados. *Softwares* de engenharia que trabalham com desenho assistido por computador fazem uso da computação gráfica para facilitar o *design* de peças mecânicas e estruturas arquitetônicas. A indústria do entretenimento utiliza a computação gráfica para criar jogos cada vez mais realistas, a indústria cinematográfica tem trabalhado com modelos virtuais que muitas vezes não podem ser distinguidos da realidade. Atualmente empresas como NVidia e Google tem dado um grande foco no desenvolvimento de tecnologias que popularizaram a realidade virtual e a realidade aumentada. Interfaces de usuário também trabalham com a computação gráfica para garantir que o usuário tenha uma melhor experiência e interatividade com o sistema operacional e programas.

O sistema gráfico atual contém dispositivos de interação com usuário (teclado, *mouse*, *joystick*), CPU (*Central Processing Unit*), GPU (*Graphics Processing Unit*) e monitor. Existem várias tecnologias diferentes de monitores, tais como LCD, Plasma, LED e OLED. Cada tecnologia varia a forma como as cores são produzidas, mas, de forma geral, todas elas trabalham com um vetor bidimensional de pixeis, em que cada píxel é uma composição das cores vermelho, verde e azul.

Em geral, um programa gráfico recebe as interações do usuário através dos periféricos e a CPU e a GPU trabalham em conjunto de forma que a CPU fornece as descrições de objetos e a GPU cria os pixeis que serão visualizados.

O *pipeline* de um sistema gráfico padrão é chamado de *pipeline* gráfico. A função dele é transformar modelos matemáticos em pixeis para serem visualizados pelo usuário. O *pipeline* gráfico realiza operações em um conjunto de objetos, no qual cada objeto é composto por primitivas gráficas. Cada primitiva contém um conjunto de vértices. A composição dessas primitivas formam a geometria dos objetos da cena. Cenas complexas, como cenários de jogos, contêm milhares ou até mesmo milhões de vértices [11].

Os principais estágios do *pipeline* gráfico são:

1. Processamento de vértices

   No primeiro estágio, os vértices são processados independentemente. As principais funções deste estágio são a transformação de coordenadas e o cálculo da cor de cada vértice. Matrizes de transformação são utilizadas para fazer concatenação de transformações que convertem a cena para o sistema de coordenadas da câmera, além de transformações para mudar escala, posição e rotação de objetos na cena.

2. Cortes e construção de primitivas

   Neste estágio, cada objeto composto por um conjunto de vértices forma uma primitiva que pode ser definida por pontos, linhas ou polígonos. Depois da formação das primitivas, é criada uma caixa de corte 3D para que se elimine tudo que se encontra no exterior da caixa. Isso é importante para que os próximos estágios do *pipeline* só processem o que está no ângulo de visão da câmera.

3. Rasterização

   Cada primitiva do estágio anterior precisa ser convertida em pixeis. Se a primitiva for um triângulo, é necessário que o interior do triângulo seja discretizado em vários fragmentos. Estes fragmentos são salvos em um *buffer* chamado de *frame buffer*. O resultado deste estágio são vários fragmentos que podem ou não tornar-se visíveis na forma de pixeis. Isso irá depender, por exemplo, de fatores como oclusão, transparência e outros.

4. Processamento de fragmentos

   No estágio final, os fragmentos sofrem transformações como mapas de textura, transformações de cores ou mistura de cores para criar efeitos translúcidos. No final desse estágio, o frame buffer é atualizado com os pixeis que serão visualizados pelo usuário.

O *pipeline* descrito acima é similar ao *pipeline* implementado pelas placas de vídeo modernas. Por muitos anos, o *pipeline* foi fixo, realizando operações básicas. Atualmente, houveram grandes avanços e parte do *pipeline* pode ser modificado e programado. As placas de vídeo em produção atualmente permitem a programação dos estágios de processamento de vértices e processamento de fragmentos. Algumas mais recentes dão a possibilidade de programação de outros estágios do

pipeline e até mesmo computação de propósito geral, utilizando a GPU não só para o processamento gráfico, mas também para aplicações altamente paralelas que se beneficiam da estrutura da placa de vídeo. A Figura 2.1 mostra os estágios do *pipeline*.

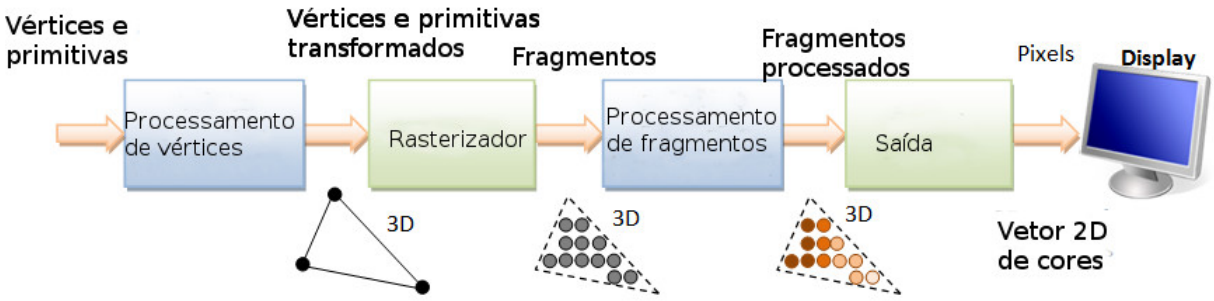


Figura 2.1: Pipeline gráfico.[2]

Duas APIs (*Application User Interface*) muito conhecidas que implementam o *pipeline* gráfico são o OpenGL (*Open Graphics Library*) e o Direct3D. O Direct3D é de propriedade da *Microsoft* e não está disponível para plataformas Linux, MacOS, Android e navegadores Web. Diferente do Direct3D o OpenGL é multiplataforma e livre. A API do OpenGL permite enviar e armazenar dados na GPU e controlar os estágios do *pipeline* para o processamento gráfico. Objetos são definidos como um conjunto de vértices. O OpenGL realiza o processamento de primitivas formadas por uma lista de vértices. Em um triângulo, por exemplo, um vértice pode ser representado no espaço cartesiano pelas coordenadas (x,y,z), um conjunto de três vértices formam um triângulo que é tratado como uma primitiva para o OpenGL. O triângulo possui várias aplicações na computação gráfica, pois é o único polígono, no qual todos os vértices sempre estarão no mesmo plano. Isto facilita o processo de rasterização e é um dos motivos das GPUs modernas não trabalharem mais com outros polígonos, somente com triângulos [12]. A Figura 2.2 mostra um objeto 3D modelado a partir de triângulos.

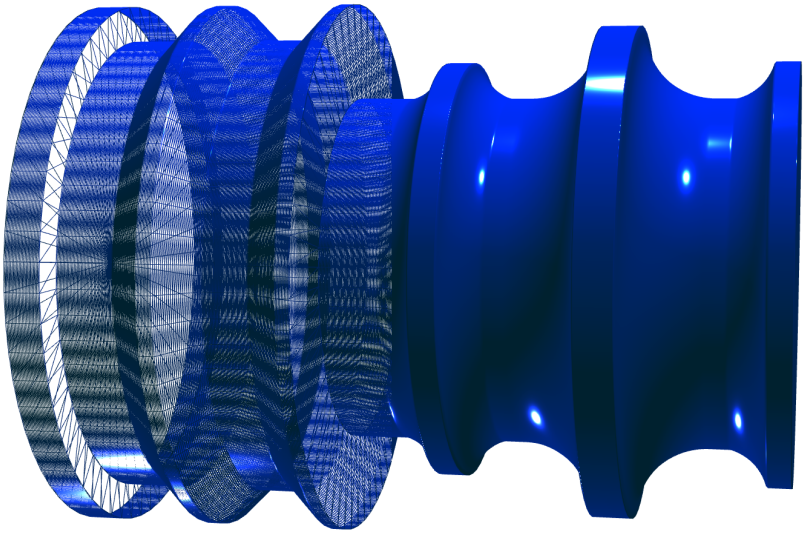


Figura 2.2: Triângulos que formam a superfície do modelo 3D.

O OpenGL não é uma biblioteca gráfica como OpenCV, é simplesmente uma API que permite controlar os estágios do *pipeline* gráfico. Apesar do foco maior em C e em C++, por questões de performance, o OpenGL pode ser utilizado em qualquer linguagem de programação.

Recentemente os *smartphones* e os sistemas embarcados ganharam um alto poder computacional, não só em termos de CPU, mas também em GPU. O OpenGL também foi portado para esse tipo de plataforma com algumas mudanças com o nome de OpenGL ES (*OpenGL for Embedded Systems*, OpenGL para sistemas embarcados). Além dos embarcados, o OpenGL também ganhou sua versão para navegadores web baseada na versão ES que é chamada de WebGL. O WebGL pode ser utilizado através do elemento canvas do HTML5 e programado em JavaScript [13]. Para todas as versões citadas, o OpenGL funciona de forma muito semelhante. Cada programa tem um contexto, podendo ter mais de um, e nesse contexto o programa funciona como uma máquina de estados, na qual é possível mudar parâmetros da máquina e mudar de estado também. Os parâmetros são configurados através da API e os estágios do *pipeline* são, em alguns casos, programados através de parâmetros configurados utilizando a API e, em outros casos, programados utilizando a linguagem de programação GLSL (*OpenGL Shading Language*). Como foi mencionado anteriormente, os estágios de processamento de vértices e processamento de fragmentos são programáveis. Isto é verdade para as versões de OpenGL superiores a 3.3 e versões do OpenGL ES, incluindo o WebGL. Versões mais recentes do OpenGL, como a 4.x, permitem a programação de outros estágios do *pipeline*, assim como a utilização da GPU para cálculos de processamento geral não envolvendo o processamento de vídeo.

A linguagem GLSL se assemelha à linguagem C em estrutura, mas possui algumas funções e variáveis pré-definidas, além de tipos de dados adicionais. Muitos dos cálculos da computação gráfica envolvem cálculos de matrizes e cálculos vetoriais. Por este motivo, a placa de vídeo possui implementação em hardware específico para acelerar esse tipo de computação. Na linguagem GLSL, é possível criar matrizes e vetores como tipo de dado fundamental. Funções padrões do GLSL permitem fazer cálculos de multiplicação de matrizes e de vetores, dentre outras operações comuns [14].

2.2 Tecnologias Web

No ambiente web, é muito comum ouvir os termos programação em *front-end* e programação *back-end*. A programação *front-end*, também chamada de *client-side*, é responsável pela interação direta com usuário. Isso aborda a coleta de dados, o pré-processamento e a interface visual do website. A programação *back-end*, também chamada de *server-side*, torna o website mais dinâmico e permite o acesso ao banco de dados e a outros dados localizados no servidor. O *front-end* e o *back-end* de um website se comunicam de forma que as requisições feitas pelo usuário são enviadas para o servidor que as processa e retorna o resultado para o *front-end*. Algumas das linguagens mais populares utilizadas no *front-end* são o HTML, Javascript e CSS. No *back-end*, são populares as linguagens como PHP, Python, Java, C#, Ruby.

Enquanto o *front-end* depende do suporte do navegador web, o *back-end* depende do servidor.

Por ser um código que roda no navegador, o código *front-end* pode ser interpretado de forma diferente em diferentes navegadores, além disso, diferentes navegadores possuem ou não suporte a diferentes funcionalidades da linguagem. A programação do *back-end* não enfrenta o mesmo problema, pois os programas rodam sempre na mesma plataforma, previamente escolhida pelo programador. Pelo fato de rodarem no servidor, qualquer linguagem de programação pode ser utilizada e o programador tem muita liberdade para fazer processamentos mais pesados ou específicos. Uma vantagem também na utilização de programas em *back-end* é que os algoritmos e programas não são acessíveis para o público. Isso garante mais proteção a códigos sensíveis e a possibilidade de esconder o código fonte dos programas.

O *front-end* ganhou muito poder com a recente chegada do HTML 5 e CSS 3. O HTML 5 trouxe novos tipos de elementos, como o canvas que permite a criação de desenhos 2D e o acesso ao contexto do WebGL, que permite acessar o *pipeline* gráfico e criar aplicações poderosas. Além do suporte à criação e à manipulação de imagens, foi incluído o suporte à criação e à manipulação de áudio, as interfaces de comunicação por sockets, a possibilidade de utilizar banco de dados com armazenamento no navegador, a possibilidade de criação de programas paralelos através de threads e muitos outros recursos. Uma consequência dessas mudanças foi o surgimento de jogos para navegador com áudio e gráficos muito realistas [15].

Na computação gráfica, existe uma grande quantidade de algoritmos que estão presentes em quase todos os projetos, alguns exemplos são os algoritmos de manipulação de câmera, transformações matriciais, simulação de modelos de luz, manipulação de materiais e texturas e etc. Atualmente, existem vários *frameworks* para computação gráfica que já fazem implementação destes algoritmos e vários outros, oferecendo aos programadores uma plataforma muito mais rápida para criar gráficos complexos e até mesmo jogos. Um *framework* muito utilizado com base no WebGL é o Three.js que é código aberto e sendo mantido desde 2010, contando com centenas de contribuidores [16][17] .

2.3 Simuladores de código-G e CNC

As primeiras máquinas de comando numérico (NC) foram construídas na década de 50 e utilizavam cartões perfurados. Naquela época, cada fabricante utilizava uma linguagem diferente para comandar as máquinas de usinagem. A Figura 2.3 mostra um exemplo de uma das primeiras máquinas controlada por comando numérico.

Em aproximadamente 1958, a linguagem chamada de código-G (G-Code) foi criada. Em fevereiro de 1980, foi criada a versão final do código-G chamado de norma RS274D. As máquinas CNC se tornaram muito populares e vários controladores foram criados. Um projeto chamado *Enhanced Machine Controller* (EMC2) atualmente chamado de LinuxCNC foi criado com o objetivo de desenvolver um controlador CNC código aberto. Com a grande utilização e popularização do micro controlador Arduino, um projeto chamado GRBL foi criado com o objetivo de possibilitar o controle de máquinas CNC utilizando Arduino [18].

Na década de 60, surgiram os primeiros softwares CAD (*Computer Aided Design*). O CAD

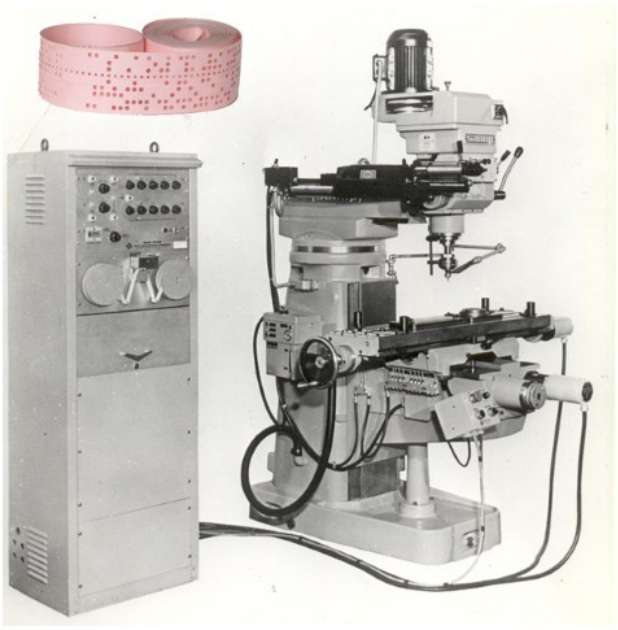


Figura 2.3: Máquina NC que utiliza cartões perfurados.[3]

substituiu o desenho em papel pelo desenho computadorizado. Depois do CAD, vieram softwares como CAM que auxiliam na manufatura. O software CAD, em conjunto com o CAM, permite que o projetista desenhe a peça e pense nos processos de manufatura, além de permitir a geração do código-G que irá comandar a máquina para produzir a peça. O principal software CAM da atualidade é o Mastercam que pode ser visto na Figura 2.4

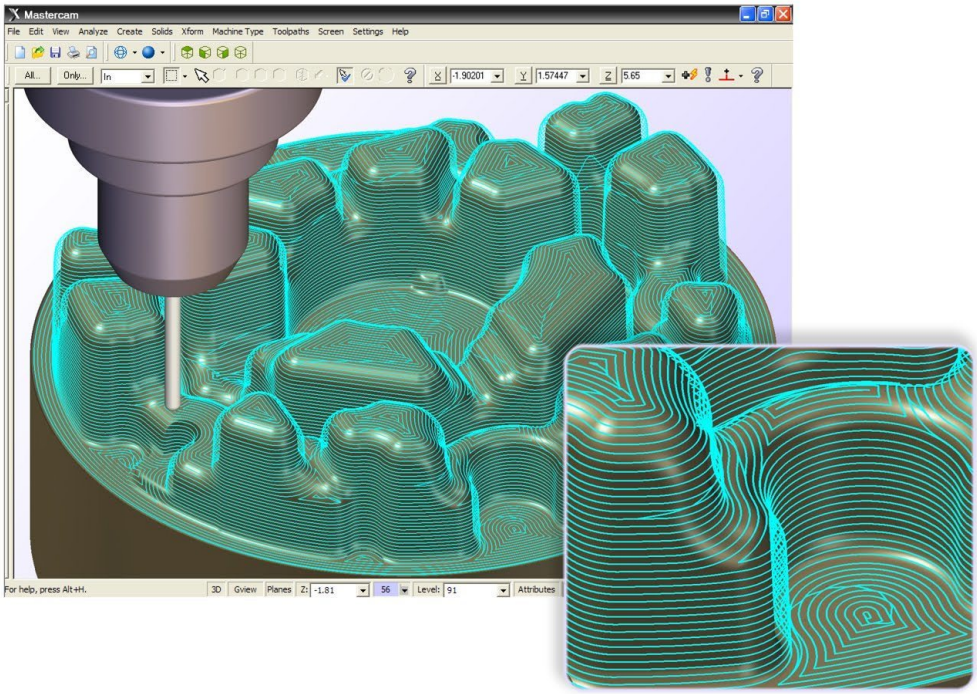


Figura 2.4: Software Mastercam - Simulação de operação de fresamento.

A Autodesk também possui software CAM que se integra com alguns dos softwares CAD mais populares. A Figura 2.5 mostra o HSM Works da Autodesk, trabalhando como um plugin no Fusion 360.

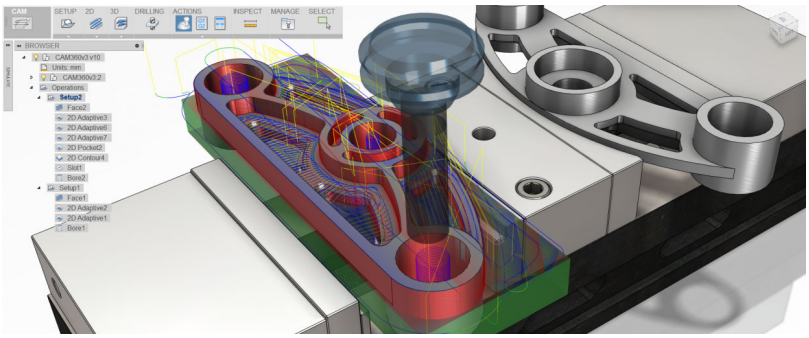


Figura 2.5: Software HSM Works integrado no Fusion 360.

No software CAM, é possível escolher a ferramenta, o tipo de usinagem e algumas opções extras para gerar o código de comando numérico. O software leva em conta o tempo de usinagem, a qualidade da usinagem, os possíveis conflitos da ferramenta com a peça e outros problemas para gerar o melhor caminho. Para o caso da impressora 3D, é necessário também verificar a possibilidade de impressão, dado que muitas vezes é necessário utilizar material de suporte.

A estrutura do código-G utilizado pelas máquinas é bem simples. O programa é dividido em blocos que são organizados um em cada linha. O bloco contém a descrição da configuração ou da função a ser executada, seguido de campos numéricos que funcionam como parâmetros para a função. O código-G possui uma grande quantidade de comandos e muitas vezes os fabricantes adicionam em seus controladores funções extras ou removem funções, no caso de controladores mais simples.

Ao utilizar o código-G, é possível controlar a ferramenta, as velocidades da máquina, o sistema de medida e muitos outros parâmetros da máquina [19]. A Figura 2.6 mostra um exemplo da estrutura do código-G. No exemplo, é possível perceber que comandos como G92, G0 e G1 recebem parâmetros que, nesses casos, são coordenadas X,Y,Z. Outros comandos funcionam mudando parâmetros da máquina.

```
G17
G21
G90
G40
G92 z50 x-10
G0 z10.000
G0 x0.000 y0.000
G0 x40.094 y235.389 z5.000
G1 z-3.000
G1 x39.594 y234.826
```

Figura 2.6: Exemplo de fragmento de código-G.

Além do CAM, existem softwares que trabalham somente com a simulação e não geram código. Um exemplo é o CNC Simulator na versão demo que possibilita escrever o código-G e simular uma operação de usinagem, além de verificar se o código foi escrito corretamente. O CAMotics é uma opção open source para simulação de máquinas CNC de 3 eixos. Em geral, os programas são capazes de simular vários tipos de controladores de máquinas CNC. Os mais completos são os simuladores para desktops, mas atualmente tem surgido alguns simuladores para navegadores web e também simuladores para *smartphones*. Uma lista destes *softwares* de simulação pode ser vista na Tabela 2.1.

	Código aberto	**Web**	**Multiplataforma**	**Simula torno**	**Simula fresadora**	**Simula impressora 3D**	**Gratuito**	**Simulação em 2D**	**Simulação em 3D**
CAMotics	Sim	Não	Sim	Não	Sim	Não	Sim	Sim	Sim
CNC Simulator	Não	Não	Não	Sim	Sim	Sim	Não	Sim	Sim
Fanuc CNC Software	Não	Não	Não	Sim	Sim	Não	Não	Sim	Sim
gCodeViewer	Sim	Sim	Sim	Não	Não	Sim	Sim	Sim	Não
Webgcode	Sim	Sim	Sim	Não	Sim	Não	Sim	Sim	Sim
Slic3r	Sim	Não	Sim	Não	Não	Sim	Sim	Sim	Sim
CNC Web Simulator	Sim	Sim	Sim	Sim	Sim	Sim	Sim	Sim	Sim

Tabela 2.1: Comparação entre alguns simuladores de código-G.

Capítulo 3

Solução proposta

Para um engenheiro, é importante ter conhecimento não só do projeto da peça, mas também ter noção de como será fabricado. Para isso, é crucial que o engenheiro projetista saiba trabalhar com um *software* CAM e também conheça a linguagem do código-G. A maioria dos simuladores existentes não são voltados ao aprendizado. Um dos programas testados que possui as melhores características para o aprendizado é o CNC Simulator, Figura 3.1, pois possui vários tipos de máquinas, sugestões de códigos e avisos de erros.

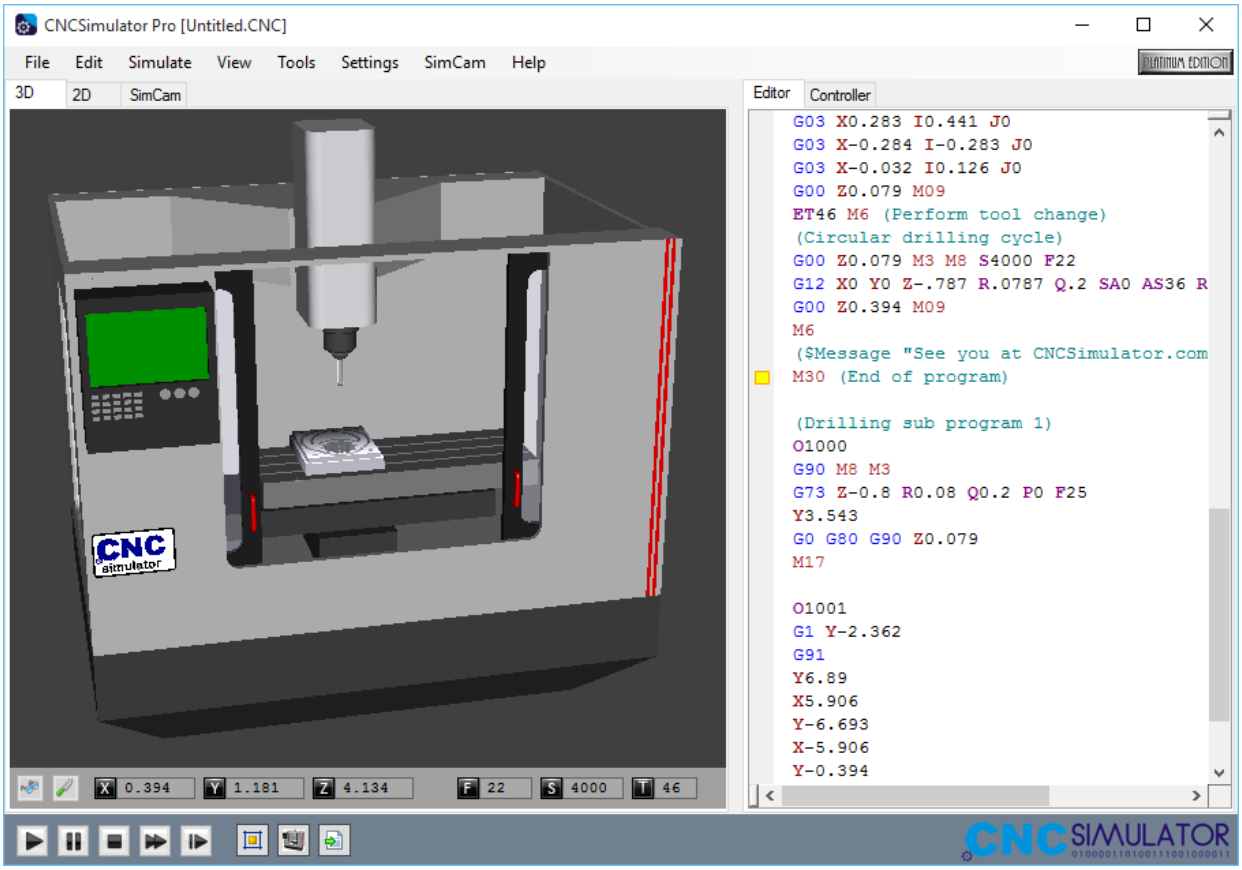


Figura 3.1: Tela do CNC Simulator simulador uma fresadora

Além do ambiente mais amigável para o aprendizado, o CNC Simulator possibilita a simulação de vários tipos de máquinas CNC de diversos fabricantes e possui um editor de códigos com cores que ajudam na leitura do código. No editor de códigos, também é possível ver sugestões e documentação da linguagem como pode ser visto na Figura 3.2.

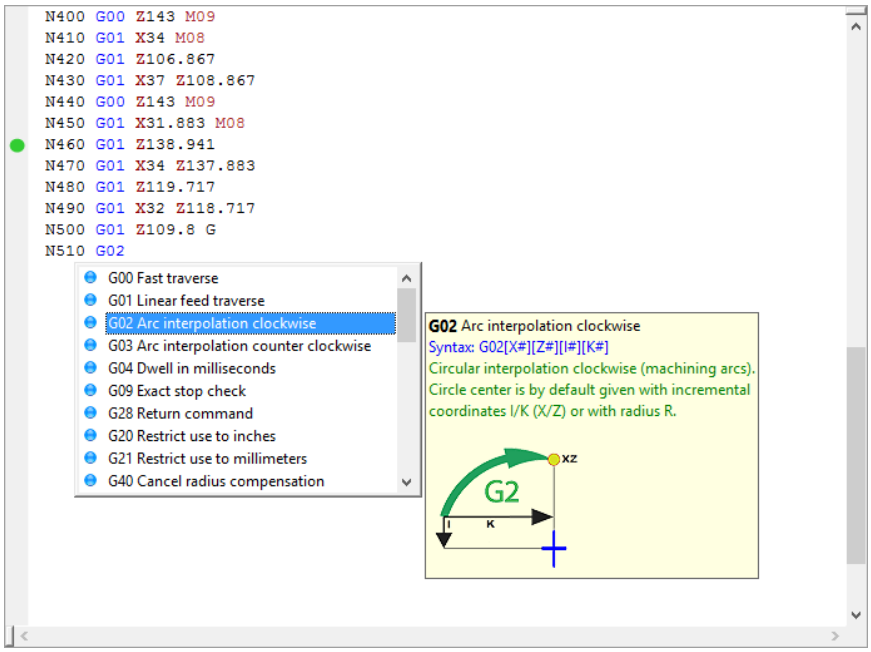


Figura 3.2: Tela do CNC Simulator - Edição de código-G

Por ser um simulador proprietário e somente para ambiente Windows, verificou-se a necessidade de criar um simulador multiplataforma e gratuito para facilitar o acesso e aprendizado sobre código-G. A solução proposta utiliza tecnologias web para criar um simulador multiplataforma que permite a utilização sem qualquer instalação. Multiplataforma inclui, não só sistemas operacionais para desktop como Windows, GNU Linux e MacOS, mas também *smartphones*, *tablets*, *smart tvs* e qualquer outra plataforma que possua um navegador web com suporte às tecnologias utilizadas. O simulador proposto foi criado em 2014 com o nome de CNC Web Simulator e foram desenvolvidas quatro versões, sendo que, na última, o código foi todo reestruturado e reescrito. A versão final pode ser vista na Figura 3.3.

O desenvolvimento foi dividido em três partes, que são:

1. Interface gráfica para interação com usuário.
2. Interpretador de código-G.
3. Engine para renderização das peças.

Cada parte foi programada e testada separadamente e, no final, tudo foi integrado para criar a versão final do simulador.

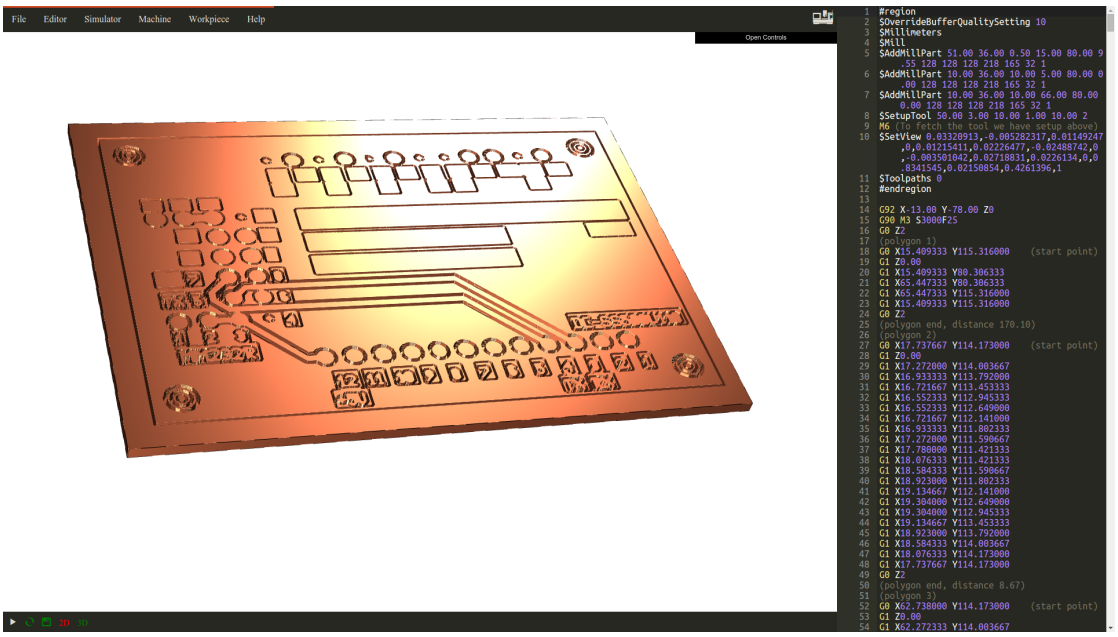


Figura 3.3: Tela do CNC Web Simulator - Simulação de placa de circuito impresso utilizando uma fresadora.

3.1 Interface gráfica

A interface gráfica foi desenvolvida utilizando as mais recentes linguagens disponíveis. Foram utilizados CSS3, Javascript, HTML5 e um *framwork* em Javascript chamado jQuery. A interface foi criada e testada nos navegadores Internet Explorer 9.0, Firefox 48.0, Chromium 52.0, Chrome Mobile 52.0 e Safari 5.1.10. A Figura 3.4 mostra a interface com o usuário, destacando cada parte que a compõe. A janela conta com 8 partes, que são:

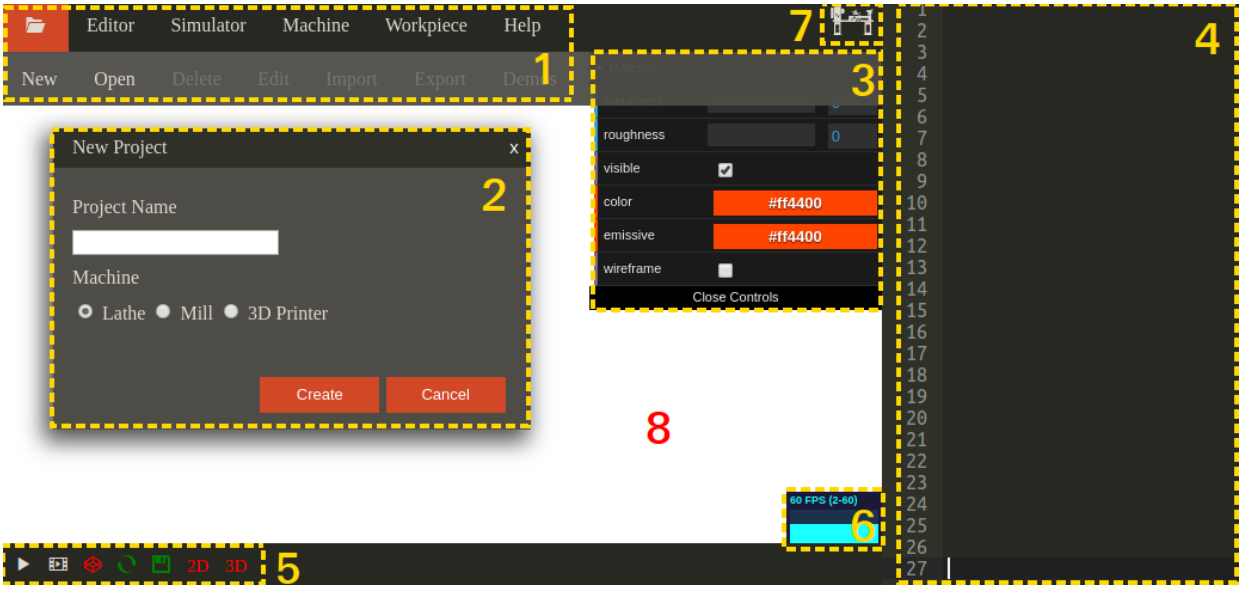


Figura 3.4: Tela do CNC Web Simulator - Partes da interface de usuário.

1. Menu do tipo *dropdown* com opções na parte superior e o submenu na parte inferior.
2. Janela aberta quando o usuário clica em alguma opção do menu.
3. Menu para configuração de opções de visualização como cores do material, luz e outros parâmetros.
4. Editor de código-G.
5. Menu com opções de simular, de salvar, de animar e de visualizar 2D ou 3D.
6. Visualizador de frames por segundo ou tempo de renderização de cada frame.
7. Ícone que mostra o tipo de máquina sendo utilizada na simulação.
8. Janela de visualização da peça e caminhos de ferramenta.

Para as plataformas móveis, a interface foi adaptada para que o editor de código-G não apareça, com isso, o usuário possui uma tela maior para visualização. Essa mudança faz com que o simulador não seja capaz de editar códigos, somente abrir e simular códigos já escritos. A Figura 3.5 mostra a versão reduzida do simulador para plataformas móveis com tela pequena.

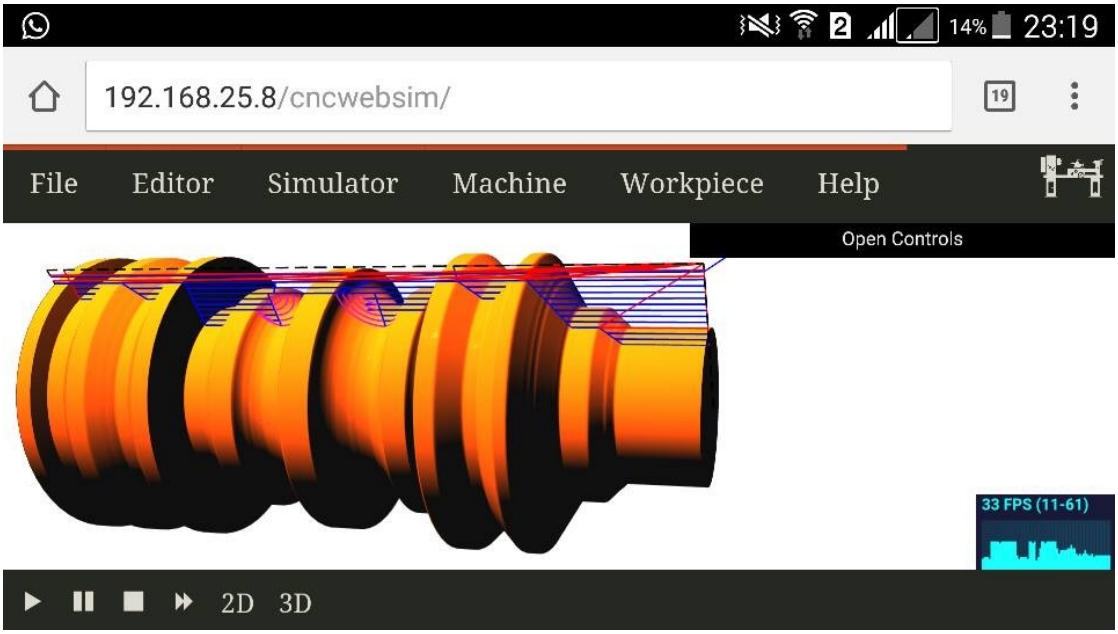


Figura 3.5: Tela do CNC Web Simulator em um celular Samsung Galaxy S5 - Navegador Google Chrome 55.

3.2 Interpretador de código-G

O interpretador de código-G implementado foi baseado no controlador Grbl [18]. O controlador Grbl é um controlador de fresadoras CNC para Arduino. Ele foi o controlador adotado como base,

pois é simples e robusto. Isso é importante, uma vez que o ambiente web é mais limitado e um controlador mais completo poderia ser inviável para implementar. Por ser um controlador de fresadoras de três eixos, algumas modificações foram feitas para aceitar códigos de tornos e impressoras 3D. O controle da máquina é o mesmo para o torno e para a fresadora com a diferença que o torno só trabalha com dois eixos. Para incluir o controle de impressoras 3D, o controlador foi modificado de forma que o código-G também aceite os comandos de extrusão de material.

O controlador Grbl produz coordenadas que são interpoladas de forma a criar pequenos segmentos de reta que são depois convertidos em sinais que vão para os motores de passos da máquina. As movimentações de ferramenta em linha reta são transformadas em pares de coordenadas com o início e o fim da reta. Os segmentos de arco são discretizados em vários segmentos de reta. A saída do controlador é uma lista de pares de coordenadas contendo o início e o fim de cada reta além de informações extras sobre qual código-G foi utilizado para gerar aquelas coordenadas.

Isso é útil para colorir as linhas com diferentes cores dependendo de quais funções do código-G foram utilizadas. A Figura 3.6 mostra um exemplo, no qual foram utilizadas funções G0 (vermelho), G1 (azul) e G2 (rosa) para movimentação de ferramenta. A interpolação em rosa se parece com um arco de circunferência mas na verdade são várias retas de tamanho muito pequeno que dão a impressão de ser uma curva.

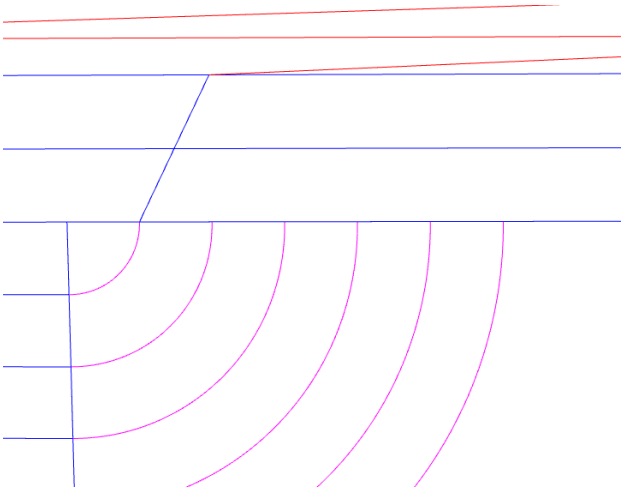


Figura 3.6: Recorte de uma simulação 2D.

3.3 Engine para renderização das peças

Como foi mencionado anteriormente, a interpretação do código-G gera uma lista com as coordenadas de início e de fim de cada reta, além de algumas informações extras sobre qual parte do código produziu aquela reta. Para todos os tipos de máquina CNC, a engine gráfica envia as coordenadas para a placa de vídeo e executa o *pipeline* gráfico que gera a visualização 2D. Na visualização 2D, é possível verificar o caminho feito pela ferramenta e quais funções do código-G foram utilizadas para gerar cada movimentação. Este estágio é bem simples e todo o processamento é realizado utilizando o *framework* Three.js mencionado em 2.2 .

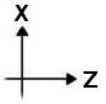
A criação das peças 3D é mais complexa, pois envolve o cálculo de material removido ou de material depositado, como é o caso da impressora 3D. Sabendo que os recursos disponíveis para uma aplicação web são reduzidos, a abordagem escolhida para tratar o problema foi utilizar a combinação de programação em Javascript com a programação da placa de vídeo. Placas de vídeo são extremamente eficientes para cálculos em paralelo. Isso significa que milhares de operações podem ser feitas simultaneamente de forma a conseguir um resultado muito mais rápido do que se fosse utilizado a programação sequencial. A programação de placas de vídeo para propósitos, que não são a criação de gráficos, é conhecida como Unidade de Processamento Gráfico de Propósito Geral ou inglês GPGPU. A técnica de GPGPU foi utilizada para discretizar a movimentação de ferramenta em pedaços muito pequenos que são calculados paralelamente de forma a gerar a peça usinada ou impressa. Como será visto mais a frente, a simulação do torno é muito similar a simulação de uma fresadora e os dois diferem da impressora 3D.

3.3.1 Simulação do torno

Para o torno, a lista de coordenadas possui as coordenadas X e Z, pois são apenas dois graus de liberdade. A lista de coordenadas é, então, carregada na memória da placa de vídeo que irá interpretar cada conjunto de coordenadas (X,Z) como um vértice. Dois vértices sequenciais formam uma reta. No primeiro estágio do OpenGL, chamado de vertex shader, são realizadas as transformações individuais para cada vértice. No final desse estágio, ocorre a clipagem dos vértices fora do cubo centrado em 0 com lados de dimensão 2. Isso significa que só vão para o próximo estágio as coordenadas entre -1.0 e 1.0 para X, Y e Z. Para renderizar a peça toda, é necessário antes fazer uma transformação nas coordenadas de forma que a peça fique dentro dos limites de -1 a 1. Isso pode ser feito da seguinte forma.

$$X' = \left(\frac{X}{R}\right) * 2 - 1$$

$$Z' = \left(\frac{Z}{L}\right) * 2 - 1$$



no qual R e L representam respectivamente o raio e comprimento da peça bruta.

Assim, todas as movimentações de ferramenta feitas fora dos limites da peça bruta são desconsideradas. No fim da execução do *pipeline* gráfico, é necessário transformar o gráfico 3D em 2D para visualização da imagem no monitor. A coordenada Z é utilizada para determinar o píxel mais próximo da tela dentre os dois píxeis possuem as mesmas coordenadas X-Y. Assim, um objeto 3D é transformado em 2D. Sabendo disso, é possível transformar as coordenadas Z em coordenadas X e as coordenadas X em coordenadas Z para que as variações no eixo radial da peça sejam vistas como variações na coordenada Z pelo OpenGL. A Figura 3.7 mostra um exemplo de uma peça com dimensões na linha pontilhada e os caminhos da ferramenta são vistos nas linhas azuis, vermelhas e rosas. É possível também ver o sistema de coordenadas da peça.

Na execução do algoritmo, as linhas fora da linha pontilhada são cortadas. Neste exemplo, somente uma linha azul é cortada. Se uma reta for traçada paralela ao eixo X, é possível obter

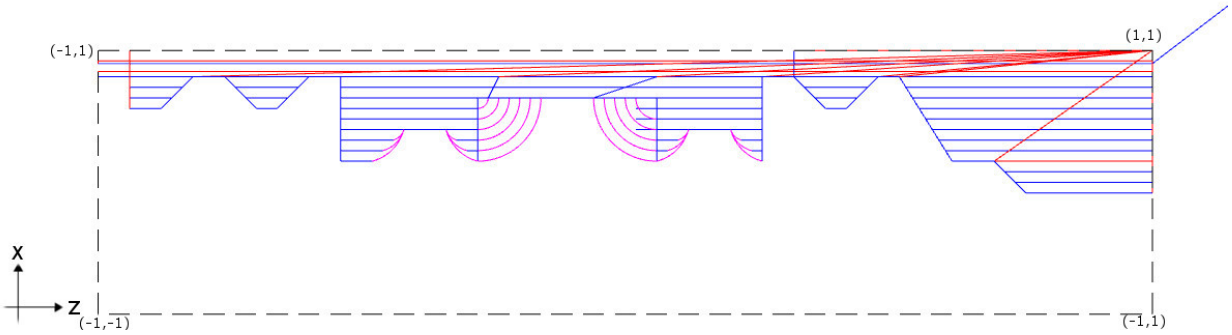


Figura 3.7: Simulação de torno.

várias coordenadas X que passam por aquela ponto. Mas para a simulação só é interessante obter a menor coordenada X para cada valor de Z. Por isso, é feita a troca da coordenada X por Z e Z por X no pipeline gráfico. Assim, o OpenGL salva a imagem final contendo somente os píxeis mais próximos, isso se traduz em píxeis referentes ao último passo da ferramenta. A imagem resultante será uma imagem de uma dimensão, pois a dimensão Z é descartada e a dimensão Y já não existia. Isso pode ser observado facilmente já que a imagem resultante seria a rotação das linhas da Figura 3.7 em 90°em torno do eixo Z. No final do *pipeline*, será gerada uma imagem de dimensão Y unitária e cada píxel possui as cores RGB das linhas desenhadas.

Só isso não é o suficiente para obter o desenho final da peça. Valores de cores não guardam nenhuma informação sobre as coordenadas X e Z da peça. Uma forma de recuperar as coordenadas dos píxeis renderizados é utilizar os canais de cores da imagem RGBA para salvar, não as cores, mas também as coordenadas dos píxeis. Com as coordenadas X e Z salvas nos canais RGBA, é possível percorrer a imagem, recuperando as coordenadas e, consequentemente, descobrindo o contorno da peça usinada. O contorno da peça da Figura 3.7 é o da Figura 3.8.

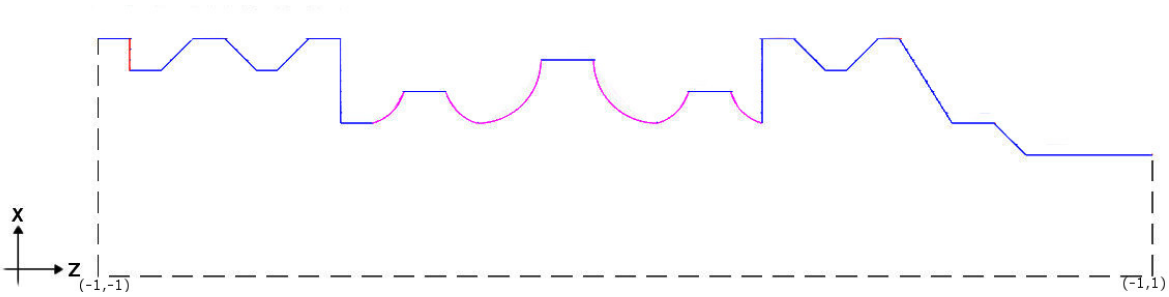


Figura 3.8: Simulação de torno - contorno da peça.

O contorno da peça é representado na forma de um *array* unidimensional de tamanho padrão 1024. O tamanho pode ser aumentado com a finalidade de melhorar a resolução no eixo Z da peça. Os valores no *array* são os valores de contorno da peça. A revolução do contorno gera os triângulos que compõem a superfície da peça. A superfície é formada por 1024 segmentos no sentido axial e 60 no sentido radial. A quantidade de triângulos na superficie é de 122.880. Diferente do sentido axial a variação na quantidade de segmentos radiais não altera a precisão da simulação, somente

a qualidade da imagem gerada. A revolução do contorno em torno do eixo Z produz um sólido 3D que é a peça usinada da Figura 3.9.

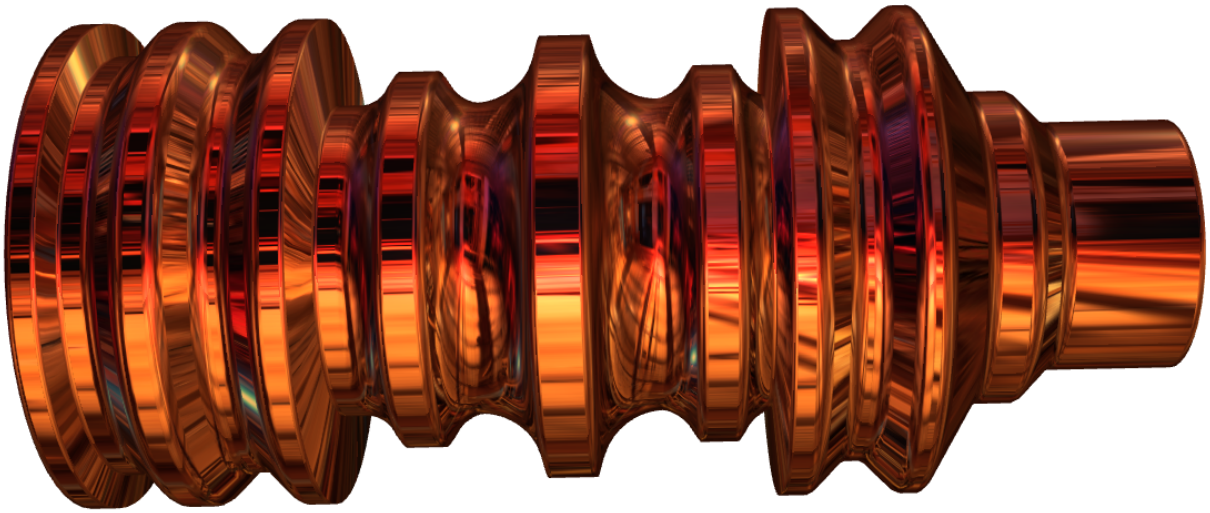


Figura 3.9: Simulação de torno - revolução do contorno da peça.

3.3.2 Simulação da fresadora

A simulação da fresadora é similar à simulação do torno. A diferença é que a fresadora possui três graus de liberdade, portanto, são utilizadas as coordenadas X, Y e Z. As coordenadas são transformadas de forma que a peça fique dentro dos limites de -1 a 1 para que seja renderizada pelo OpenGL. Na fresadora, pode acontecer da ferramenta passar várias vezes pela mesma coordenada X e Y com profundidades diferentes no eixo Z. Por isso, é utilizado o plano X,Y como plano da imagem e, no eixo Z, o OpenGL faz a escolha dos píxeis mais próximos da face inferior da peça. De forma similar ao torno, o algoritmo gera o caminho da ferramenta que irá usinar a peça. O resultado é uma imagem bidimensional, na qual os canais RGBA guardam informações X, Y e Z, mas o raio da ferramenta de corte da fresadora não foi especificado ainda. Primeiro as coordenadas X e Y são armazenadas nos canais RG e BA da imagem na forma de 16 bits. Depois a imagem é renderizada novamente e a coordenada Z é armazenada nos canais RG. Para usinar a peça com um raio específico, é realizado outra execução do pipeline. Dessa vez, as coordenadas do passo anterior são utilizadas para criar quadrados. Cada vértice se transforma no centro de um quadrado no mesmo plano X,Y e com lados de mesma dimensão que o diâmetro do raio da broca da fresadora. Na execução do último estágio do pipeline, no qual são criados os píxeis, são descartados os píxeis exteriores a circunferência inscrita no quadrado. O resultado é a imagem 2D, na qual os canais RGBA guardam as coordenadas X, Y e Z da peça usinada com uma broca específica. Para transformar isso em uma peça 3D, é utilizado a técnica de mapa de altura onde uma imagem bidimensional é utilizada para armazenar valores de elevação da superfície e isso é utilizado para gerar uma malha poligonal com a superfície em 3D. Na técnica de mapa de altura as coordenadas X e Y têm um espaçamento regular e a altura Z é dada pelos valores contidos na imagem. A desvantagem é a menor resolução nas coordenadas X e Y da peça que só pode ser

melhorada com um aumento na resolução do mapa de altura. Um exemplo de imagem gerada mostrado na Figura 3.10.

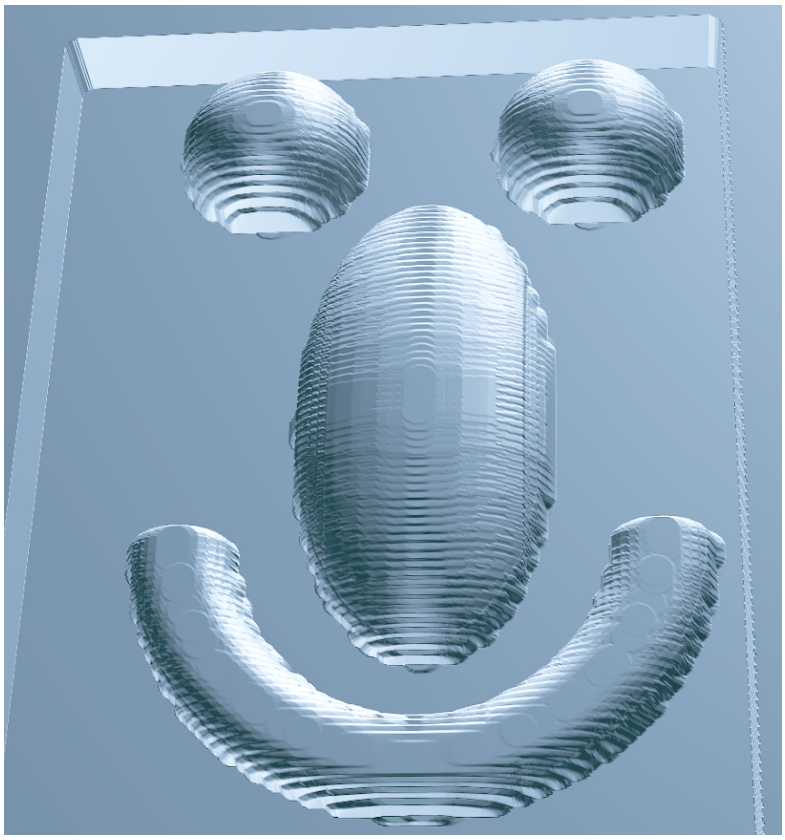


Figura 3.10: Simulação de fresadora - mapa de altura do caminho da ferramenta de corte.

3.3.3 Simulação de impressora 3D

A simulação da impressão 3D é bem mais simples. O interpretador de código-G gera a lista com as coordenadas somente para as movimentações de ferramenta de extrusão, em que há depósito de material. Depois de passar pelo interpretador, todas as movimentações são interpoladas para movimentações em linha reta. Sendo assim, a única coisa que deve ser feita é transformar as retas bidimensionais em retas com volume. Para dar volume às retas e para simular o filamento da impressão 3D, são gerados 4 planos paralelos à reta com rotação de 90° entre si. A Figura 3.11 mostra um exemplo de filamento utilizado para renderizar as peças impressas.

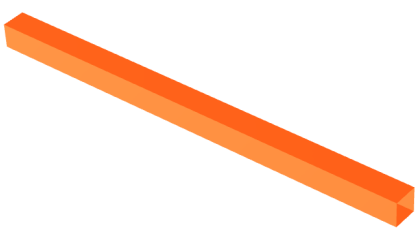


Figura 3.11: Filamento da impressora 3D usado na simulação.

O ideal para a simulação é utilizar o filamento como um tubo cilíndrico e não quadrado. O problema disso seria a quantidade de polígonos na imagem. Cada fragmento de filamento possui oito polígonos. A mudança de um tubo quadrado para um cilíndrico aumentaria consideravelmente a quantidade de polígonos na imagem final, deixando a renderização lenta e inviável. A aproximação do filamento para um tubo quadrado tem gerado resultados plausíveis e suficientemente rápidos. O problema maior da simulação para impressões 3D tem sido a quantidade de polígonos da imagem. As impressões 3D possuem milhares a alguns milhões de polígonos para serem renderizados. O simulador não remove as faces redundantes pois é computacionalmente muito caro. A Figura 3.12 mostra uma peça renderizada pelo simulador.

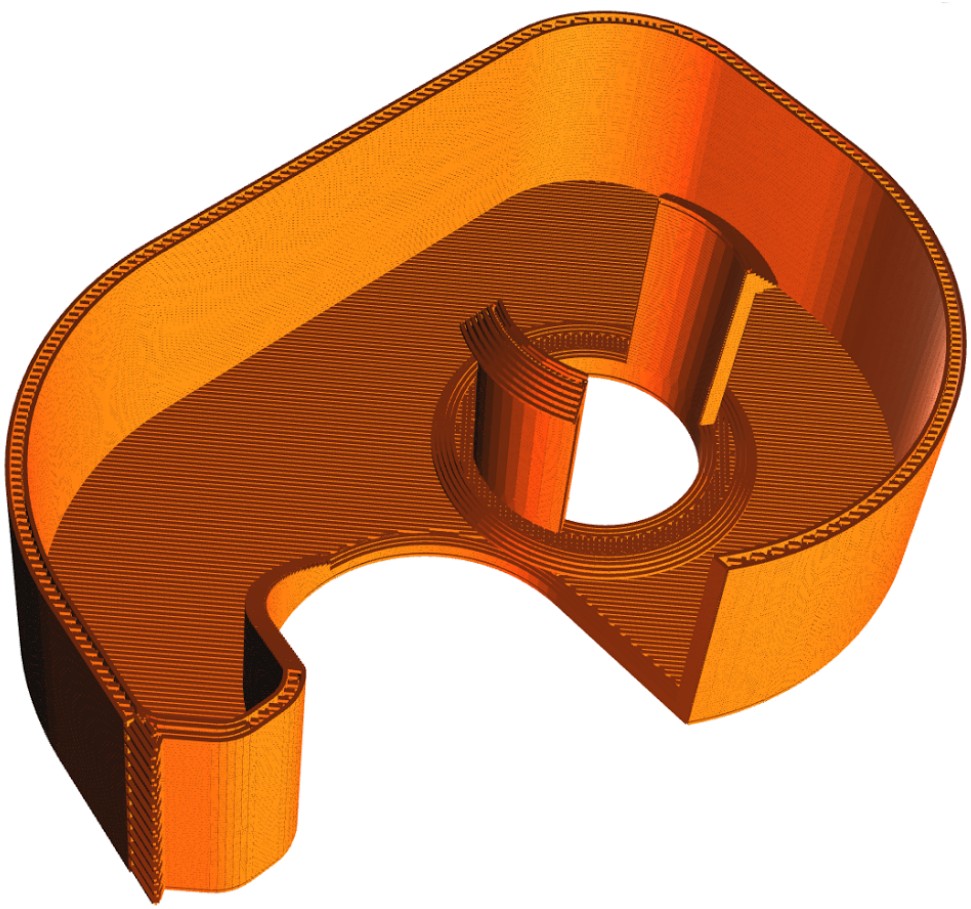


Figura 3.12: Impressão 3D de um dispensador de fitas.

3.3.4 Organização do código

O código foi desenvolvido utilizando a ferramenta Git para controle de versão. O *software* de controle de versão possibilita trabalhar colaborativamente com outros desenvolvedores, além de fazer cópias de diferentes versões do código ao longo do desenvolvimento. Com isso, é possível voltar em versões passadas para o caso de alguma modificação que deve ser revertida ou para comparação de modificações no código. Todo código fonte foi hospedado no site GitHub para que qualquer usuário tenha acesso ao código e à versão demonstrativa do simulador. O *link* para

o código fonte e para a versão demo do simulador é encontrado no seguinte endereço `https://github.com/filipecaixeta/cncwebsim`.

O código fonte do simulador possui aproximadamente cinco mil linhas e está organizando na seguinte estrutura:

1. index.html - Arquivo HTML por onde o simulador é executado.
2. js - Pasta que contém todos os códigos Javascript.
   - 2.1 libs - Pasta com bibliotecas e dependências.
     - 2.1.1 ace - Pasta com códigos Javascript do editor de códigos-G utilizado como padrão pelo CNC Web Simulator.
     - 2.1.2 dat.gui.js - Interface gráfica para mudança de variáveis em Javascript. A biblioteca possibilita que as variáveis de cores e luzes sejam modificadas para renderização.
     - 2.1.3 jquery.js - Biblioteca para facilitar a captura de eventos e para gerenciar janelas e elementos da interface de usuário.
     - 2.1.4 lz-string - Biblioteca projetada para compressão de grandes quantidades de texto em navegadores web. É utilizada no simulador para comprimir código-G que será armazenado *off-line*.
   - 2.2 storage.js - Responsável pela criação da estrutura dos projetos, armazenamento *off-line*, compressão e descompressão de código-G.
   - 2.3 project.js - Possui estruturas base para criação de máquinas padrão, peça de usinagem padrão e código-G de exemplo.
   - 2.4 editor.js - Cria o editor de códigos e configura eventos para mudança de código.
   - 2.5 shaders.js - Contém todos os códigos na linguagem GLSL que são compilados mais tarde e executados na placa de vídeo.
   - 2.6 ui.js - Gerência eventos gerados por interação com usuário e criação de janelas da interface gráfica.
   - 2.7 renderer.js - Renderiza as peças em 3D, adiciona iluminação e cor.
   - 2.8 parser.js - Faz o pré-processamento do código-G e organiza em as linhas para serem interpretadas.
   - 2.9 interpreter.js - Interpreta o código-G, gerando uma lista de coordenadas e de caminhos de ferramenta.
   - 2.10 motion.js - Gerencia a thread que interpreta o código-G.
   - 2.11 motionWorker.js - Transforma código-G em coordenadas de movimentação de ferramenta. Todo o código é executado em uma *thread* diferente da principal.
   - 2.12 machine.js - Base para todos os tipos de máquina CNC. Também responsável pela renderização 2D de todos os tipos de máquina.
   - 2.13 lathe.js - Implementa funções específicas do torno, e cria o modelo 3D da peça.

2.14 mill.js - Implementa funções específicas da fresadora, e cria o modelo 3D da peça.

2.15 printer.js - Implementa funções específicas da impressora 3D, e cria o modelo 3D da peça.

3. css - Pasta que contém códigos CSS.

3.1 ui.css - CSS principal da interface gráfica.

3.2 jquery-ui.structure.css - CSS necessário para a criação de janelas do programa.

3.3 contIcon.css - Alguns ícones do simulador foram criados utilizando fontes modificadas. Este arquivo contém o CSS necessário para que os ícones sejam visualizados.

Javascript é uma linguagem que permite diversos tipos de paradigmas de programação. Neste projeto, maior parte do código foi desenvolvido utilizando o paradigma chamado Prototype. Este paradigma é utilizado em situações nas quais se espera alta performance e cópias de objetos através de clonagem. O diagrama de sequências na Figura 3.13 mostra a sequência de chamadas de funções desde a entrada do código-G à peça final em 2D e 3D.

O anexo I possui um manual de usuário para utilização do simulador.

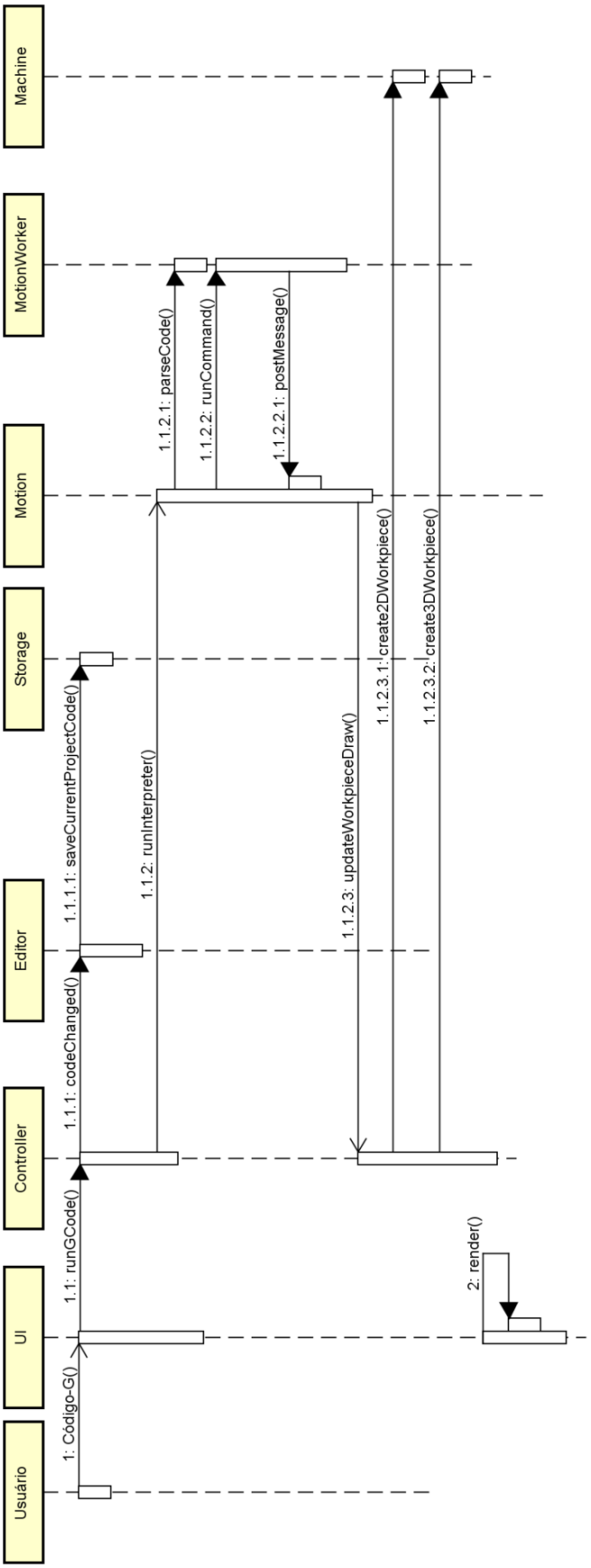


Figura 3.13: Diagrama de sequências da execução do código-G

Capítulo 4

Resultados

O projeto foi hospedado no site GitHub de modo que qualquer um tenha acesso ao simulador [20]. Durante todo o tempo, foram coletados dados utilizando o Google Analytics para ter estatísticas sobre onde o simulador está sendo utilizado e os tipos de usuários. Um dos objetivos do simulador era resolver o problema da falta de simuladores para plataformas diferentes do Windows. Apesar da não divulgação do simulador na Internet, o simulador contou com uma grande quantidade de acessos. A partir da Tabela 4.1, percebe-se que houve uma alta procura por usuários que não utilizam sistema operacional Windows. Grande parte dos acessos são de computadores Desktops, mas há também alguns acessos em dispositivos móveis.

Sistema Operacional	Sessões	Porcentagem das sessões
MAC	291	31,94%
Windows	286	31,39%
Linux	214	23,49%
Android	80	8,78%
iOS	26	2,85%
Outros	14	1,54%

Tabela 4.1: Sistema Operacionais mais utilizados para acessar o simulador.

Foi observado também que 58,9% dos acessos foram no Brasil e 41,1% foram de outros países. A maioria dos acessos foram da Universidade de Brasília, onde alguns estudantes têm testado o simulador por dois semestres. Estudantes foram orientados a testar o simulador, reportar problemas e dar sugestões para melhorar a produtividade e facilidade de uso. Para analisar os resultados, alguns alunos desenvolveram o plano de usinagem e código que foram testados em dois *softwares*, CNCSimulator Pro e o simulador proposto por este trabalho chamado de CNC Web Simulator.

A Figura 4.1 mostra a simulação de torneamento de uma peça utilizando o CNCSimulator Pro. É possível observar claramente que o simulador divide a peça em vários cilindros e calcula a passagem da ferramenta por cada um deles. O número de divisões é pequeno e isso faz com que seja perceptível a olho nu. A simulação utilizando o CNC Web Simulator na Figura 4.2 faz

1024 divisões e no final faz interpolações lineares entre elas para que seja imperceptível e o usuário consiga visualizar a peça que parece ser contínua. A peça usinada por um torno CNC pode ser vista na Figura 4.3.

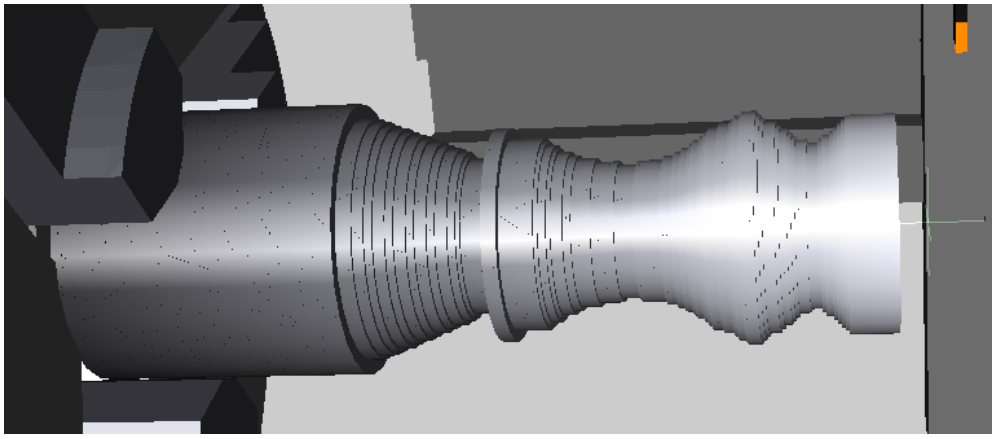


Figura 4.1: Simulação de torno utilizando CNCSimulator Pro.

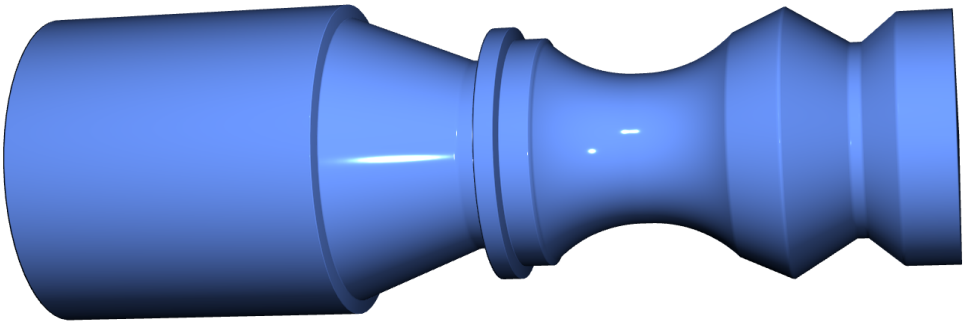


Figura 4.2: Simulação de torno utilizando CNC Web Simulator.

Figura 4.3: Peça usinada por um torno CNC (Cortesia de Guilherme Caetano Gonçalves).

As Figuras 4.4, 4.5 e 4.6 também apresentam o mesmo resultado, no entanto, foram utilizados códigos diferentes.

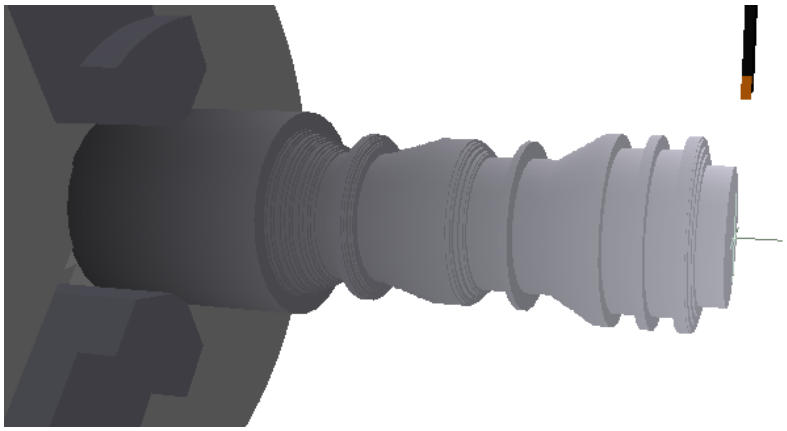


Figura 4.4: Simulação de torno utilizando CNCSimulator Pro.

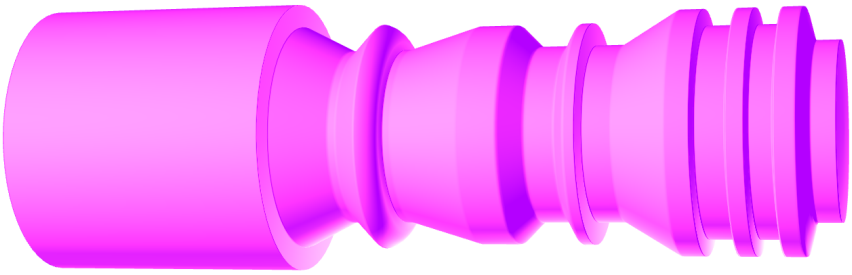


Figura 4.5: Simulação de torno utilizando CNC Web Simulator.

Figura 4.6: Peça usinada por um torno CNC (Cortesia de Marcos Pereira).

A simulação da fresadora é mais complexa que a simulação do torno e renderiza um pouco mais de meio milhão de polígonos. É possível ajustar a qualidade para renderizar um número maior de polígonos, podendo chegar a alguns milhões. Como pode ser visto na Figura 4.7, o CNCSimulator Pro possui resolução menor na renderização, que acaba gerando contornos mais serrilhados e imprecisos. A Figura 4.8 mostra a simulação no CNC Web Simulator, na qual uma melhora na simulação é observada. A peça usinada por uma fresadora CNC pode ser vista na Figura 4.9.

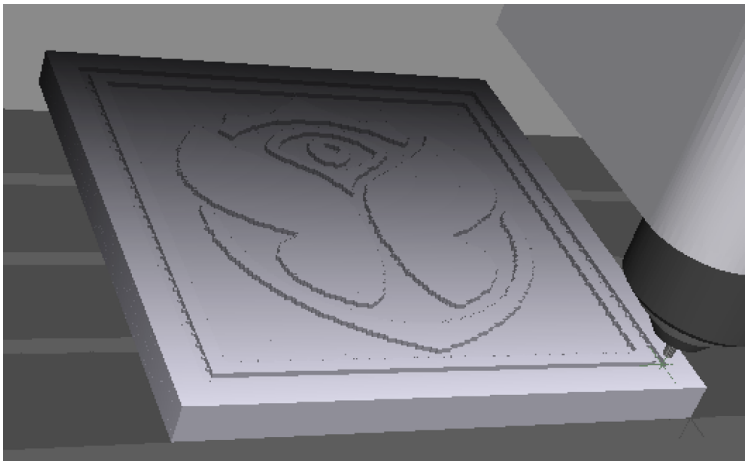


Figura 4.7: Simulação de fresadora utilizando CNCSimulator Pro.

Figura 4.8: Simulação de fresadora utilizando CNC Web Simulator.

Figura 4.9: Peça usinada por uma fresadora CNC (Cortesia de Renan Costa).

As Figuras 4.10, 4.11 e 4.12 possibilitam verificar que o efeito de bordas serrilhadas está presente no CNC Web Simulator em menor quantidade. Além disso, a possibilidade de ajustar a iluminação e a cor da peça facilitam a visualização.

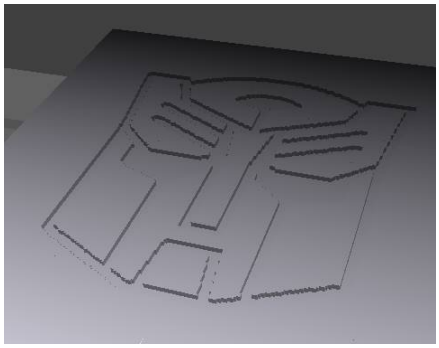


Figura 4.10: Simulação de fresadora utilizando CNCSimulator Pro.

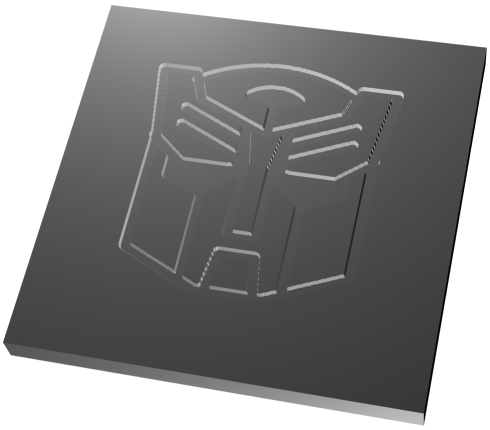


Figura 4.11: Simulação de fresadora utilizando CNC Web Simulator.

Figura 4.12: Peça usinada por uma fresadora CNC (Cortesia de Marcos Pereira).

As simulações de impressões 3D foram feitas utilizando um software chamado Slic3r para transformar modelos 3D em código G e fazer simulações. A Figura 4.13 mostra uma simulação feita no Slic3r. Na figura 4.14, é possível observar a simulação feita pelo CNC Web Simulator. A impressão 3D pode ser vista na Figura 4.15. Uma diferença entre as duas simulações é que o software Slic3r simula o depósito de material como curvas contínuas, enquanto que o CNC

Web Simulator faz a simulação utilizando segmentos discretos sem conexão um com o outro. A simulação de segmentos contínuos é mais realista, mas consome uma quantidade maior de recursos, principalmente placa de vídeo. A simulação da Figura 4.14 utilizou pouco mais de dois milhões de triângulos. Para fazer as curvas contínuas seriam necessários o dobro de triângulos, portanto seria inviável para uma aplicação web. A Figura 4.16 e 4.17 mostra uma parte dos olhos ampliadas, possibilitando comprar a diferença entre os segmentos contínuos e discretos.

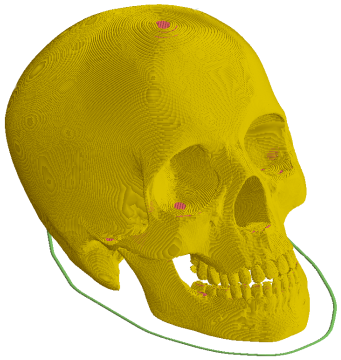


Figura 4.13: Simulação de impressora 3D utilizando Slic3r.

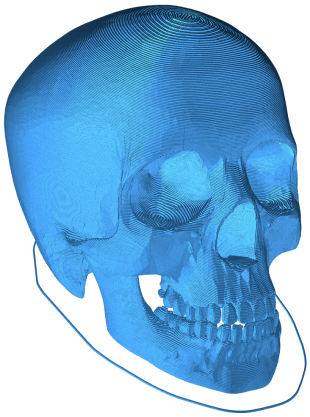


Figura 4.14: Simulação de impressora 3D utilizando CNC Web Simulator.

Figura 4.15: Peça impressa por uma impressora 3D MakerBot [4].

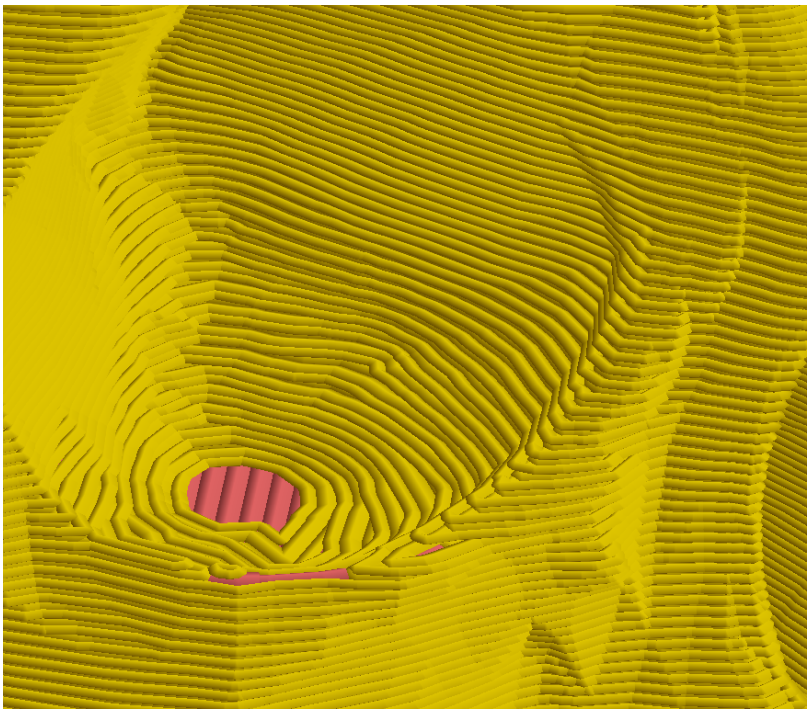


Figura 4.16: Simulação de impressora 3D utilizando Slic3r com zoom.

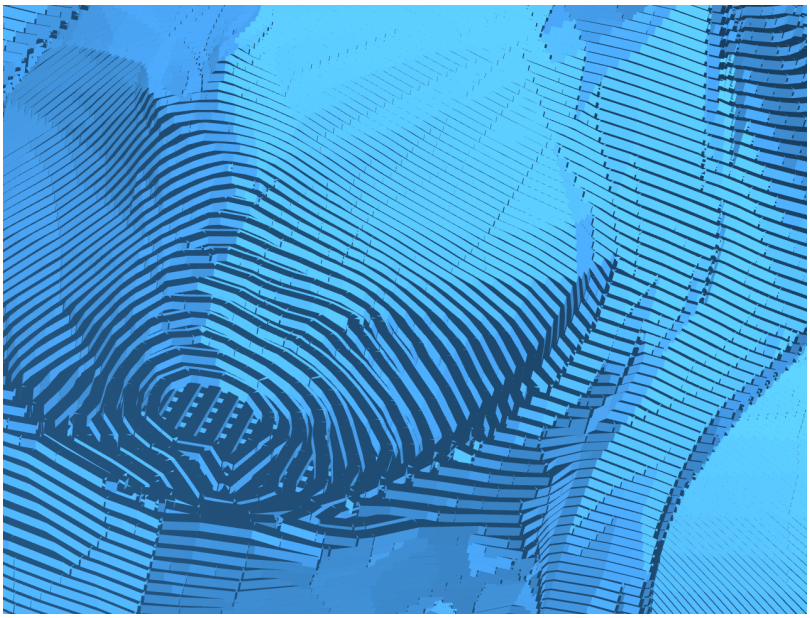


Figura 4.17: Simulação de impressora 3D utilizando CNC Web Simulator com zoom.

Na Figura 4.18, observa-se que a geometria circular utilizada nos filamentos da simulação produz um efeito na imagem que distorce a luz na superfície do material. Isso gera os círculos vistos principalmente na superfície entre o olho e a orelha esquerda. Os círculos estão presentes também na pata traseira e algumas distorções da luz na lateral do corpo do elefante. Na Figura 4.19, a reflexão da luz na superfície é mais regular, o que produz um resultado mais realista da peça impressa como mostrado na Figura 4.20.

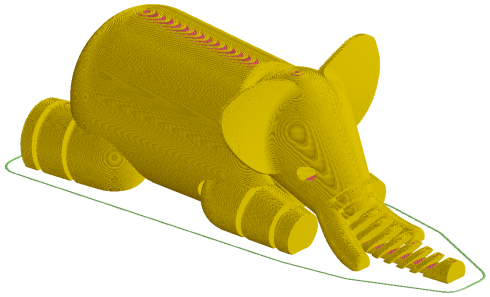


Figura 4.18: Simulação de impressora 3D utilizando Slic3r.

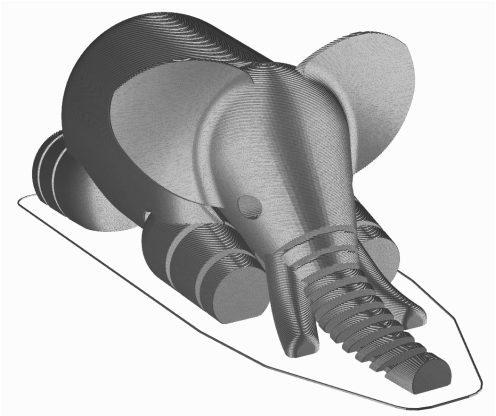


Figura 4.19: Simulação de impressora 3D utilizando CNC Web Simulator.

Figura 4.20: Peça impressa por uma impressora 3D MakerBot (Cortesia de Vitor Rezende).

A Figura 4.21 mostra um caso extremo em que a superfície e a falta de iluminação dificultam a visualização de detalhes da peça. Na Figura 4.22, a mão possui detalhes como veias, unhas e dobras da pele. A possibilidade de modificar a iluminação e propriedades do material são pontos fortes que contribuem para a melhor visualização das peças. A peça impressa é vista na Figura 4.23.

Figura 4.21: Simulação de impressora 3D utilizando Slic3r.

Figura 4.22: Simulação de impressora 3D utilizando CNC Web Simulator.

Figura 4.23: Peça impressa por uma impressora 3D MakerBot.

Capítulo 5

Conclusões e trabalhos futuros

O projeto foi bem-sucedido em criar um simulador que fosse multiplataforma e de fácil utilização. A plataforma escolhida foram os navegadores web, na qual foi possível criar um *software* compatível com computadores, *smartphones* e *tables*. Foram encontrados alguns problemas de compatibilidade, pois as tecnologias utilizadas são muito recentes e não são totalmente implementadas em todos os navegadores.

O simulador foi projetado pensando na mais recente tecnologia disponível. Por esse motivo, alguns navegadores desatualizados não são capazes de suportar todas as funcionalidades. Ate mesmo navedagores atualizados podem não funcionar corretamente se alguma funcionalizadade não estiver ativada. Foi observado que drives de vídeo mal instalados também podem fazer com que o simulador não funcione. O problema será resolvido ao longo do tempo, quando as tecnologias se tornarem mais populares e difundidas nos navegadores web. Uma forma de resolver o problema seria simular e renderizar a peça por *software* eliminando a necessidade do OpenGL ou driver para placa de video.

O simulador foi testado por alguns alunos da Universidade de Brasília e se mostrou promissor. O projeto obteve aceitação de uma grande quantidade de alunos que optou por utilizar o CNC Web Simulator no lugar de outros simuladores. A facilidade em utilizar o simulador, a velocidade e a qualidade das simulações foram pontos fortes.

As simulações feitas foram comparadas com simulações de outros simuladores e com peças usinadas ou impressas. Desta forma, foi possível validar o simulador. Apesar de ser mais rápido e produzir peças com mais qualidade do que os simuladores utilizados na comparação, alguns casos mais avançados não foram testados. O simulador foi implementado pensando em simulações básicas de máquinas simples. Não foram abordadas simulações de centros de usinagem, máquinas com troca de ferramentas ou funções especiais de diferentes fabricantes. O CNCSimulator Pro utilizado nas comparações é capaz de simular uma grande quantidade de máquinas, utilizando ferramentas mais complexas, códigos-G especiais e alguns outros auxiliares. O CNC Web Simulator foi implementado de forma a ser um simulador mais simplificado, abordando somente o básico por questões de limitações de recursos dos navegadores web.

Uma deficiência nas simulações foi a incapacidade de fazer animações para mostrar a peça

sendo usinada. Este foi um fator que levou muitos a optar por outros simuladores. Esse tipo de animação foi possível de implementar nas impressoras 3D, por se tratar de depósito de material. Nas máquinas de usinagem, o cálculo de verificação de qual parte da peça foi usinada iria deixar a animação muito lenta e inviável. A animação da usinagem do torno e da fresadora iria resultar em um tráfego enorme de dados entre a CPU e GPU e muito processamento tanto na CPU, quanto GPU que seriam inviáveis para códigos longos. Para resolver o problema das animações, foi implementado a animação do caminho de ferramenta em 2D para todos os tipos de máquinas.

Para trabalhos futuros, é interessante adicionar funcionalidades extras, como a opção de clicar em uma linha e visualizar ela destacada na renderização 2D. Mais opções para criar geometria de ferramentas é algo importante para as simulações. Alguns simuladores também permite usinar em peças importadas e não somente em peças brutas no formato cilíndrico ou paralelepipédico. Nas plataformas *mobile*, não é conveniente programar em código-G, por isso uma funcionalidade a ser implementada seria a possibilidade de salvar projetos no servidor para que o usuário possa logar em outros computadores e acessar o código. Algo interessante também seria integrar ferramentas CAM com o simulador. Existe uma infinidade de software CAM de código aberto. Uma integração do CNC Web Simulator com um deles possibilitaria realizar as funcionalidades de um CAM no servidor e comunicar com o simulador para devolver os resultados e simular o modelo 3D da peça.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] 3D BLUE MEDIA. *MakerBot Replicator 5th Generation.* Disponível em: http://www.3dbluemedia.com/3d-printing-equipment/3d-printers/makerbot-replicator-5th-generation.html. Acesso em: 02/12/2016.

[2] NTU. *3D Graphics Rendering Pipeline.* Disponível em: https://www.ntu.edu.sg/home/ehchua/programming/opengl/CG_BasicsTheory.html. Acesso em: 02/12/2016.

[3] PROTOPTIMUS. *Máquina Fresadora NC controlado por Fita Perfurada.* Disponível em: http://www.protoptimus.com.br/maquinas-cnc-historia-comando-numerico-computadorizado/. Acesso em: 02/12/2016.

[4] CULTS 3D. *Skull.* Disponível em: https://cults3d.com/en/various/to-make-or-not-to-make. Acesso em: 06/11/2016.

[5] YASH MACHINE. *History of Lathe from Beginning of Machine Tool Invention.* Disponível em: http://www.yashmachine.com/blog/history-of-lathe-from-beginning-of-machine-tool-invention/. Acesso em: 29/07/2016.

[6] ROMI. *ROMI T SERIES.* Disponível em: http://www.romi.com/en/produtos/romi-t-series/. Acesso em: 01/12/2016.

[7] BANKA. *What Is Milling Machine ? History Of Milling Machine.* Disponível em: https://ravimachines.com/what-is-milling-machine-history-of-milling-machine/. Acesso em: 29/07/2016.

[8] PRODUCTIVE TOOLS. *Plano Milling.* Disponível em: http://www.productivetools.co.in/index.php/facilities/milling-plano-milling. Acesso em: 01/12/2016.

[9] KRAMER, T. R.; PROCTOR, F. M.; MESSINA, E. The nist rs274/ngc interpreter- version 3. *National Institute of Standards and Technology*, 2000.

[10] 3D PRINTING INDUSTRY. *History of 3D Printing.* Disponível em: https://3dprintingindustry.com/3d-printing-basics-free-beginners-guide/history/. Acesso em: 29/07/2016.

[11] ANGEL, E.; SHREINER, D. *Interactive Computer Graphics: A Top-Down Approach with Shader-Based OpenGL.* 6th. ed. USA: Addison-Wesley Publishing Company, 2011. ISBN 0132545233, 9780132545235.

[12] KHRONOS GROUP. *Rendering Pipeline Overview.* Disponível em: `https://www.opengl.org/wiki/Rendering_Pipeline_Overview`. Acesso em: 29/07/2016.

[13] KHRONOS GROUP. *WebGL Specification.* Disponível em: `https://www.khronos.org/registry/webgl/specs/1.0/`. Acesso em: 29/07/2016.

[14] WEBGL FUNDAMENTALS. *WebGL Shaders and GLSL.* Disponível em: `http://webglfundamentals.org/webgl/lessons/webgl-shaders-and-glsl.html`. Acesso em: 29/07/2016.

[15] W3. *HTML5.* Disponível em: `https://www.w3.org/TR/html5/`. Acesso em: 29/07/2016.

[16] THREE.JS. *Three.js.* Https://threejs.org/.

[17] DIRKSEN, J. *Learning Three.js: The JavaScript 3D Library for WebGL.* 1st. ed. UK: Packt Publishing, 2013.

[18] SIMEN SVALE SKOGSRUD. *Grbl.* Https://github.com/grbl/grbl.

[19] KRAMER, T. R.; PROCTOR, F. M.; MESSINA, E. *The NIST RS274/NGC Interpreter-Version 3.*

[20] CNC Web Simulator. Disponível em: `https://github.com/filipecaixeta/cncwebsim`.

ANEXOS

I. MANUAL PARA UTILIZAR O SIMULADOR

O simulador está disponível no seguinte link `www.filipecaixeta.github.io/cncwebsim`. Ao acessar o site, a seguinte tela pode ser vista. Por padrão, o simulador abre um projeto do tipo simulação de torno com uma peça padrão. A partir do segundo acesso ao site, será aberto o último projeto ativo. A cada minuto, qualquer modificação no código é salva. Desta forma, o simulador pode recuperar o código em caso de problemas, como fechar o navegador por engano ou algum problema no computador. Na barra inferior, há o símbolo de um disquete que, quando está verde, significa que todas as modificações foram salvas. Para forçar o simulador a salvar o código, o usuário pode clicar no disquete e salvar manualmente.

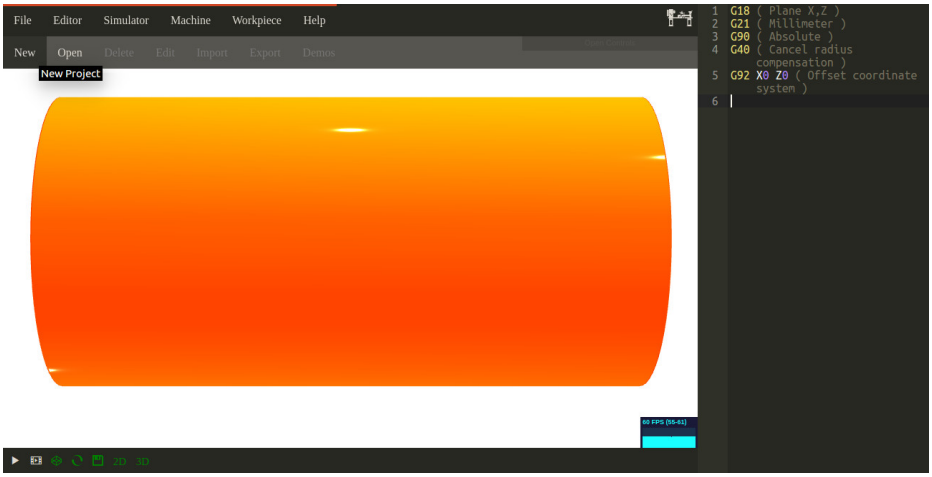


Figura I.1: CNC Web Simulator - Tela inicial

Para abrir um projeto existente, o usuário deve clicar em **File** e, em seguida, em **Open**. Uma janela com projetos anteriores é aberta. Para criar um projeto novo, o usuário deve clicar em **File** e, em seguida, em **New**.

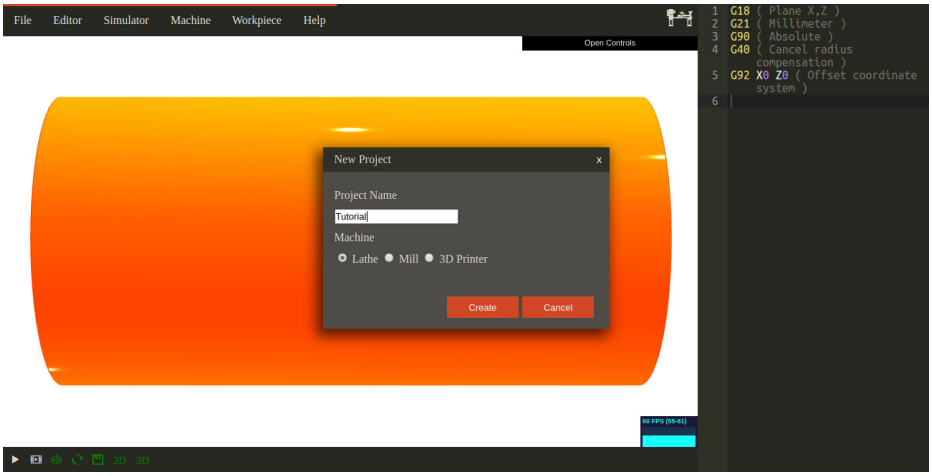


Figura I.2: CNC Web Simulator - Novo projeto

Sempre que um novo projeto é criado, o simulador muda o tipo de simulação para o tipo do projeto. É possível modificar o tipo de simulação em **Machine**, escolhendo um tipo de máquina diferente.

Após criar o projeto, é necessário modificar as dimensões da peça bruta ou os parâmetros do filamento da impressora 3D. Isso é feito em **Workpiece** -» **Dimensions**.

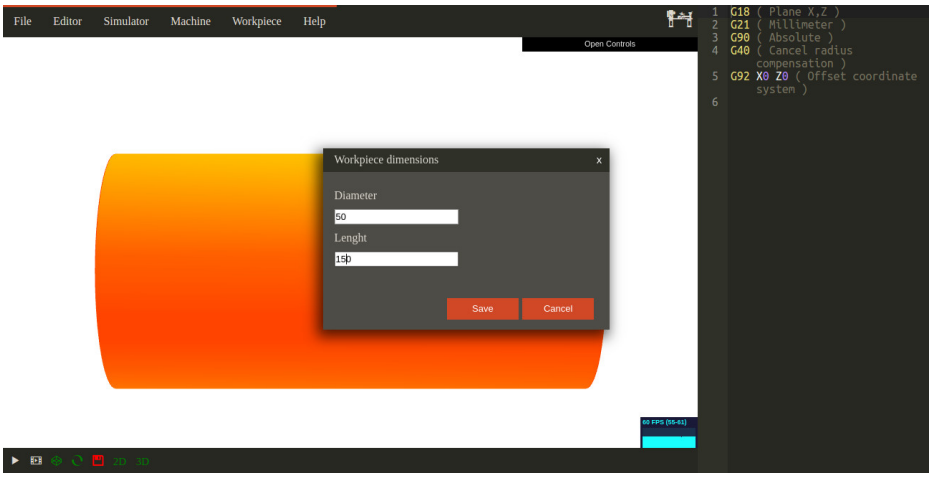


Figura I.3: CNC Web Simulator - Dimensões da peça

Além de modificar as dimensões da peça, é necessário modificar os parâmetros da ferramenta em **Machine** -» **Tool**.

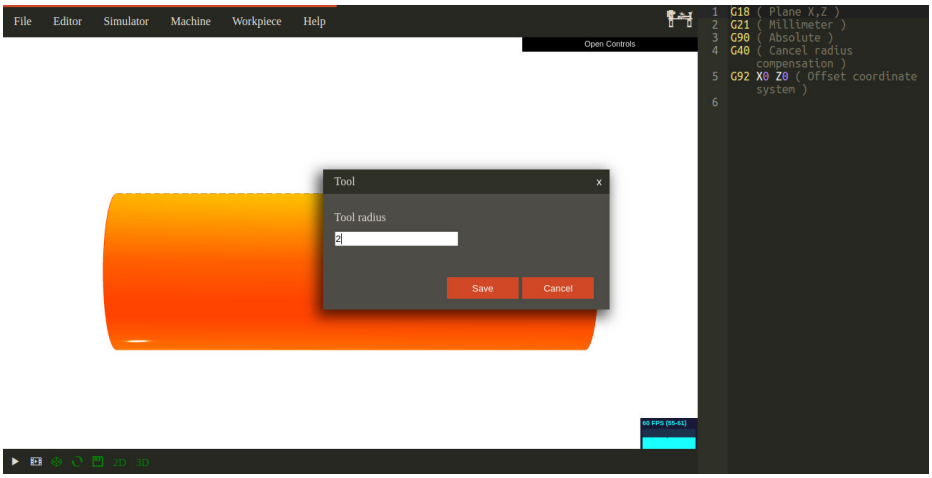


Figura I.4: CNC Web Simulator - Ferramenta

Agora, o usuário já pode começar a programação da máquina, utilizando código-G. O editor de códigos do simulador é configurado para atualizar a renderização sempre que o código é modificado. Para casos em que o código é muito grande, excedendo algumas milhares de linhas, é recomendável desativar a execução automática no ícone **auto run** ao lado do disquete. Para executar, o código no modo manual basta dar **play**.

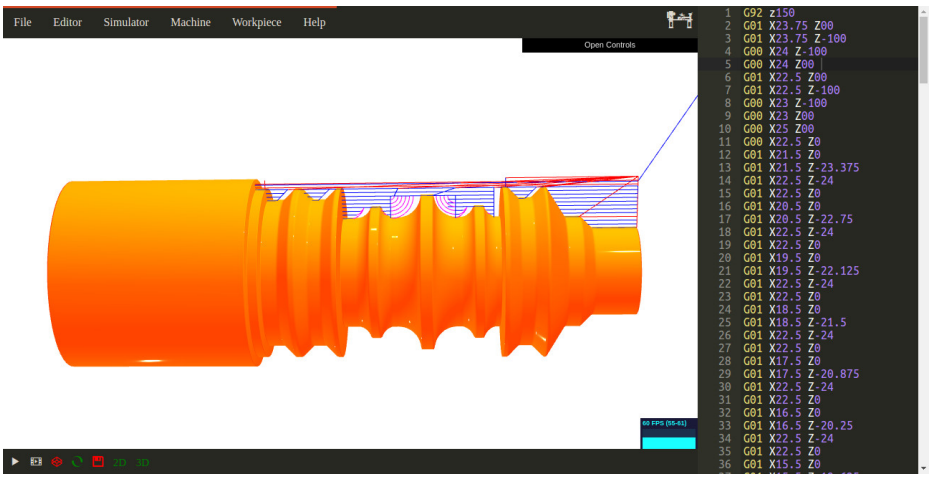


Figura I.5: CNC Web Simulator - Simulação 2D e 3D

A execução do código permite a visualização do frame da peça, da simulação 2D e 3D. No menu inferior, é possível ocultar alguma das visualizações. Desativar o 3D pode aumentar consideravelmente a velocidade do simulador, principalmente em computadores menos potentes.

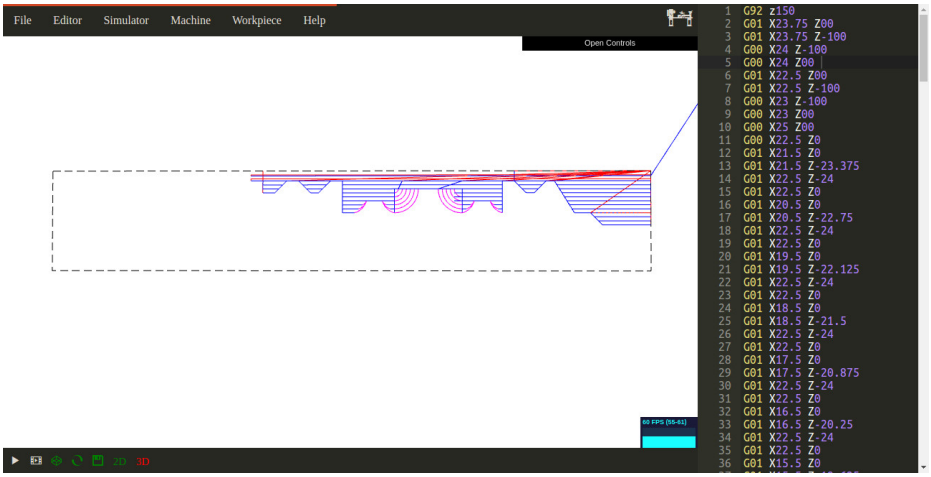


Figura I.6: CNC Web Simulator - Simulação 2D

Nem sempre é fácil visualizar uma peça 3D, dependendo da cor e iluminação. Uma boa iluminação e escolha de cores pode facilitar a visualização de detalhes na peça. Para isso o simulador possui um menu no canto superior direito, onde o usuário pode alterar algumas propriedades do material como a cor, o modo como o material reage a luz e as propriedades da superfície do material.

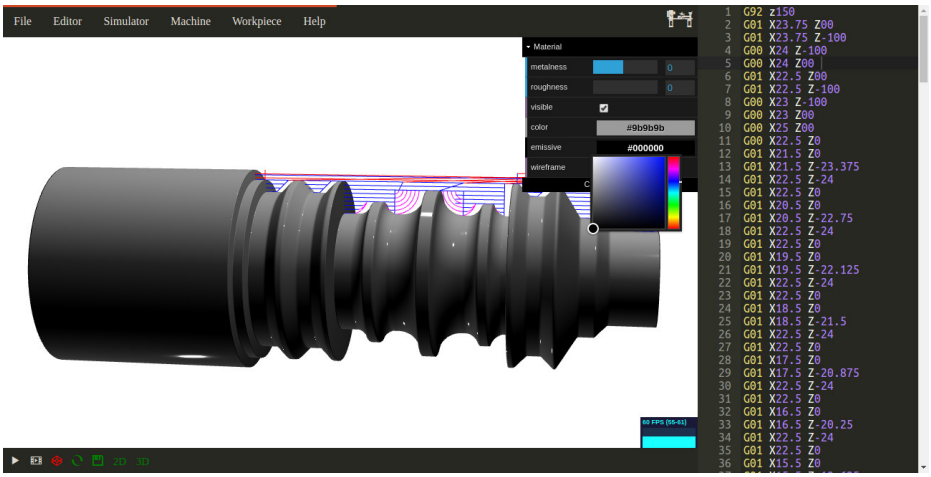


Figura I.7: CNC Web Simulator - Propriedades do material