CENTRO UNIVERSITÁRIO LUTERANO DE PALMAS

Recredenciado pela Portaria Ministerial nº 1.162, de 13/10/16, D.O.U. nº 198, de 14/10/2016
AELBRA EDUCAÇÃO SUPERIOR - GRADUAÇÃO E PÓS-GRADUAÇÃO S.A.

Igor Valadares Queiroz

Simulador online de fractais

Palmas – TO

2021

Igor Valadares Queiroz

Simulador online de fractais

Trabalho de Conclusão de Curso (TCC) II elaborado e apresentado como requisito parcial para obtenção do título de bacharel em Ciência da Computação pelo Centro Universitário Luterano de Palmas (CEULP/ULBRA).

Orientador: Prof. M.e Fabiano Fagundes.

Palmas – TO

2021

Igor Valadares Queiroz

Simulador online de fractais

Trabalho de Conclusão de Curso (TCC) II elaborado e apresentado como requisito parcial para obtenção do título de bacharel em Ciência da Computação pelo Centro Universitário Luterano de Palmas (CEULP/ULBRA).

Orientador: Prof. M.e Fabiano Fagundes.

Aprovado em: _____/_____/_____

BANCA EXAMINADORA

_____

Prof. M.e Fabiano Fagundes

Orientador

Centro Universitário Luterano de Palmas – CEULP

_____

Prof.a. M.a Madianita Bogo Marioti

Centro Universitário Luterano de Palmas – CEULP

_____

Prof.a Esp. Fernanda Pereira Gomes

Centro Universitário Luterano de Palmas – CEULP

Palmas – TO

2021

AGRADECIMENTOS

Primeiramente gostaria de agradecer a Deus, por possibilitar tantas coisas boas em minha vida. Agradeço aos meus pais por sempre se esforçarem para me proporcionar uma boa educação.

Agradeço a todos os professores do CEULP/ULBRA por sempre se esforçarem para nos entregar algo de extrema qualidade em especial ao meu orientador Fabiano Fagundes. Agradeço aos meus amigos Bruno Fortaleza, Nalberthy Sousa, Emanoel Mendes, Jheymerson Lira, Danilo Saraiva, Antônio Carlos, Ricardo que foram fundamentais no processo de desenvolvimento, sempre me apoiando e me dando forças nos momentos em que mais precisei de ajuda.

RESUMO

QUEIROZ, Igor Valadares. **Simulador online de fractais**. 2021. 22f. Trabalho de Conclusão de Curso (Graduação) – Ciência da Computação, Centro Universitário Luterano de Palmas, Palmas/TO, 2021.

A geometria fractal surgiu a partir de quando a geometria Euclidiana não era mais o suficiente para a representação de objetos da natureza, por sua forma simplista de representação. Então, o matemático Benoit B. Mandelbrot associa os fractais a objetos e fenômenos da natureza. Os fractais têm uma característica marcante que é a sua autossimilaridade, que seria a referência com o todo, não importando o nível de ampliação. A partir desse contexto, foi idealizado um sistema que foi possível simular fractais clássicos em que parâmetros modificados pelos usuários refletem em tempo real nos fractais sendo gerados. Foi utilizada a linguagem de programação web para a sua implementação e posteriormente disponibilizado o sistema de forma online.

LISTA DE FIGURAS

SUMÁRIO

1 INTRODUÇÃO

“Nas últimas décadas aconteceram investigações cujo tema central foi a construção e o estudo de entidades geométricas, tais entidades (ou objetos) foram chamadas fractais pelo seu criador, Benoit Mandelbrot” (BARBOSA, 2002, p.9), sendo que tais fractais possuem uma característica que é a auto-similaridade que, segundo Aguilera (2008, p. 1), “uma forma que se repete em si mesma, de maneira semelhante – não necessariamente exatamente igual – e independente de proporção, se diz auto-similar”, sendo esta característica a que mais chama atenção nos fractais.

Antes de se falar no termo fractal abordado por Mandelbrot já existiam alguns trabalhos matemáticos relacionados a área da geometria fractal como, por exemplo: Curva de Koch, Triângulo de Sierpinski, Curva de Peano que são classificados como fractais clássicos.

Segundo Rossetto (2012, p. 14), “o desenvolvimento de aplicações para web evidenciou uma significativa expansão nos últimos anos”. Esta expansão proporcionou a criação de ferramentas de simulação web que auxiliam diversas áreas. No caso dos estudos dos fractais pode-se ter a mesma possibilidade, pois o processo de visualização dos fractais torna-se fácil e acessível a todos possibilitando ques seja possível, a partir de uma aplicação web ter um contato mais próximo no processo de geração dos fractais e tornando um importante artefato tecnológico.

A utilização de um simulador proporciona uma experiência única e influencia diretamente no que está sendo visto. Quando se trata de fractais trabalha-se diretamente com formas geométricas complexas e, a partir do momento que se tem um mecanismo que proporciona um contato visual por meio da tecnologia, acaba motivando ainda mais o usuário que está entrando em contato com os fractais. E, isso tem uma grande importância, pois proporciona ao usuário um ambiente em que este pode ter acesso a experimentação do comportamento e uma melhor compreensão dos fractais a partir da modificação de parâmetros e visualização dos fractais sendo gerados.

O presente trabalho apresenta o desenvolvimento de uma ferramenta capaz de simular um ambiente online de fractais em que o usuário pode realizar modificações nos parâmetros assim refletindo nos fractais em tempo real. E os resultados do processo de desenvolvimento, são apresentados na seção 4.

2 REFERENCIAL TEÓRICO

2.1 FRACTAIS

Por muito tempo a geometria euclidiana foi utilizada para descrever a natureza que nos cerca, porém com o passar dos tempos os termos de linhas, círculos, cubos já não conseguia explicar a complexidade que existe. Benoît Mandelbrot, o pai da geometria Fractal, percebeu tal dificuldade e propôs a utilização dela para descrever os objetos reais, tais como árvores, nuvens, rios e crosta terrestre. O termo Fractal surgiu em 1975 pelo seu iniciador Benoît Mandelbrot, em que o termo fractal vem do adjetivo latino que significa Fractus, do verbo “Frangere”, que significa quebrar.

Um fractal é visto como um objeto que possui a autossimilaridade, não importando o nível de ampliação, com efeito de simetria com o todo, cada parte replicando a semelhança com a estrutura do todo (ADDISON 1997, p. 2). Tal similaridade pode ser vista na Figura 1.

Figura 1 - Similaridade

Fonte: Marília Gomes Negri (2014)

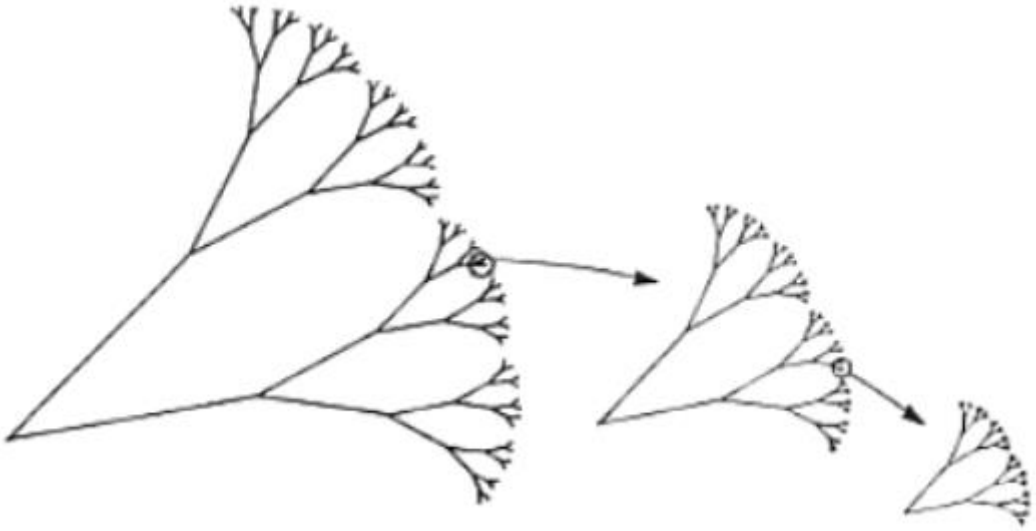
Pode-se perceber que cada ramo da folhagem é semelhante a toda a folhagem e assim por diante, ou seja, apresenta uma autossimilaridade exata, e à medida que se move para a parte superior pode-se ver uma folhagem cada vez menor que a anterior.

A aplicação da geometria fractal é bastante diversificada e com um forte apelo visual que chama bastante atenção, e tem diversas aplicações nas ciências e tecnologias, como por exemplo: na otimização de espaço, medicina, biologia, agricultura e entre outros.

As principais propriedades que compõem os fractais são a sua autossemelhança e a sua complexidade infinita. A autossemelhança representa uma

semelhança com o todo o fractal, podendo ser exata ou aproximada, mas sempre mantendo características semelhantes ao todo, não importando o nível que ampliamos o objeto, como pode ser visto na Figura 2.

Figura 2 - Autossemelhança



Fonte: Assis et al. (2008)

A Figura 2 retrata a imagem de uma árvore fractal bifurcada representando a ideia de autossimilaridade falada anteriormente.

De acordo com Negri (2014, p.18), a complexidade infinita “refere-se ao fato do processo de geração de uma Figura, definida como sendo um fractal, ser recursivo, tendo um número infinito de iterações”, ou seja, independentemente da medida que é ampliado o objeto fractal nunca terá uma imagem completa, pois ela estará sendo gerada infinitamente.

2.2 CLASSIFICAÇÃO

Os fractais podem ser classificados em três categorias, segundo Alves (2007, p.35) “Sistema de funções iteradas, Fractais definidos por relação de recorrência e Fractais aleatórios”, que serão explicados e exemplificados a seguir.

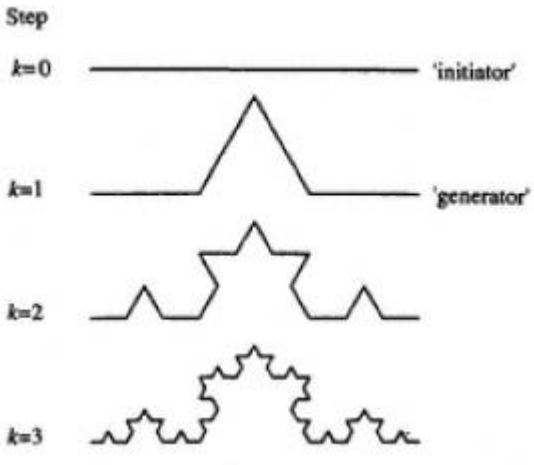
2.2.1 SISTEMA DE FUNÇÕES ITERADAS

A categoria sistema de funções iteradas possui uma regra fixa de substituição geométrica e funções iterativas, são conhecidos como fractal determinístico. Os fractais gerados por tais funções têm a autossimilaridade exata, assim, o fractal é

idêntico em todas as escalas (RABAY, 2013). Um fractal bastante conhecido desta categoria é a Curva e Ilha de Koch, que pode ser visto na Figura 3.

A curva de Koch foi criada por Helge von Koch, e em 1904 ele realiza uma publicação de seus estudos, e seu processo de construção pode ser visto na Figura 3.

Figura 3 - Construção da curva de Koch



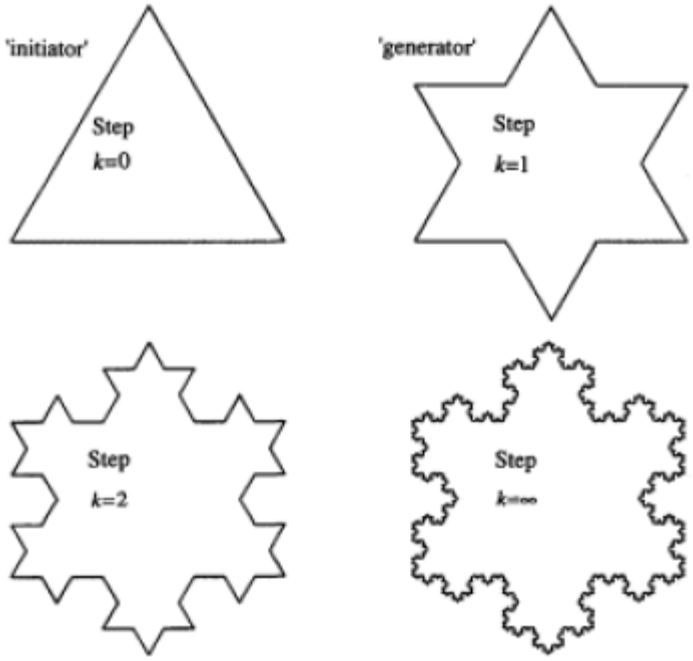
Fonte: Addison (1997, p.17)

Sua construção é feita com base em três etapas, como:

- etapa 1: a construção começa com uma única linha de reta;
- etapa 2: utilizar o segmento anterior dívida em três partes iguais removendo a parte do meio, e colocando duas partes de um triângulo equilátero sem a sua base, assim a curva de Koch não possui nenhuma tangente em nenhum ponto;
- etapa 3: consiste em repetir o processo anterior de forma iterativamente.

Ao final de todo o processo resulta a autossimilaridade, bastando escolher em qualquer fase um segmento a ser substituído e observar que ele irá gerar e seguir uma curva semelhante à curva completa de Koch (BARBOSA, 2005, p.32). Por sua vez, a Ilha de Koch ou *Snowflake* (Floco de neve) é composta por três curvas de Koch e sua construção pode ser vista na Figura 4.

Figura 4 - Ilha de koch



Fonte: Addison (1997)

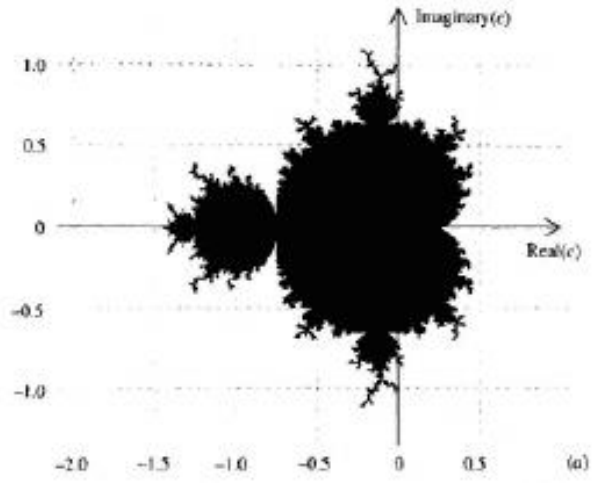
A Figura 4 apresenta imagens criadas de acordo com o número de iterações que foi repassada à função, e a medida em que este número é incrementado a ilha de Koch sofre alterações em sua proporção, tornando impossível saber o tamanho de seu comprimento.

2.2.2 FRACTAIS DEFINIDOS POR RELAÇÃO DE RECORRÊNCIA

Os fractais por relação de recorrência não possuem autossimilaridade, este tipo de fractais possuem uma forma livre e “apresenta pequenas cópias do fractal inteiro de maneira distorcida ou degenerada por isso não são auto similares exatamente” (RABAY, 2013, p.27). O fractal de Mandelbrot é um exemplo de fractal definido por relação de recorrência.

O Conjunto de Mandelbrot está entre os mais retratados nos documentos científicos. Suas propriedades matemáticas estão longe de serem totalmente compreendidas e a partir de um computador e poucas linhas de código de computador pode ser gerado um fractal por Falconer (2013, p.75). A Figura 5 mostra um exemplo do conjunto de Mandelbrot.

Figura 5 - Conjunto de Mandelbrot



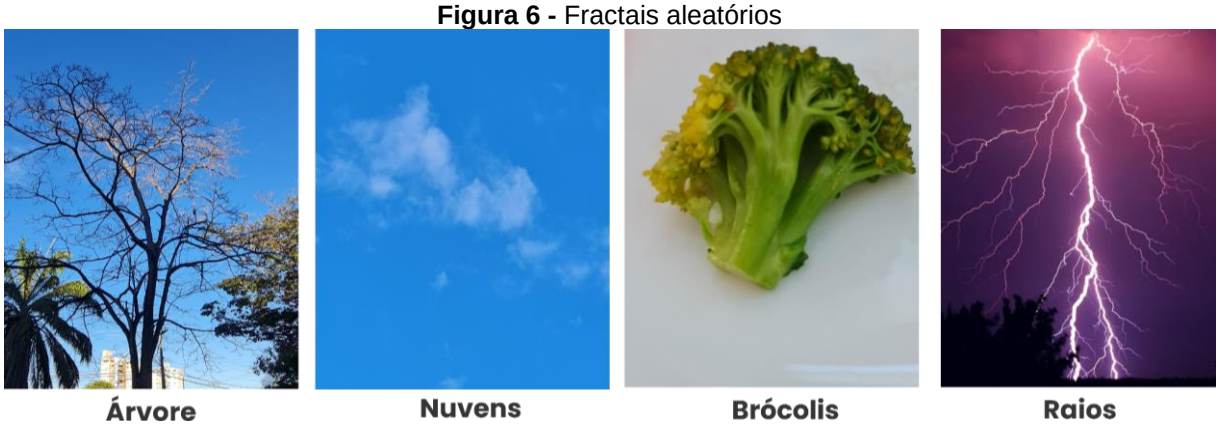
Fonte: Addison (1997, p. 109)

A Figura 5 mostra o conjunto de Mandelbrot, onde a geração de um conjunto de Mandelbrot só é possível através de recursos computacionais, para que seja garantido a autossimilaridade. Para tal representação é feita com base em conjuntos e funções matemáticas em um plano complexo, tal plano é utilizado para representar graficamente números complexos e imaginários.

2.2.3 FRACTAIS ALEATÓRIOS

Os fractais aleatórios, ou fractais naturais, são estatisticamente auto similares, onde cada parte do fractal tem as mesmas propriedades estaticamente do todo (ADDISON, 1997, p.27). Exemplos de fractais aleatórios podem ser vistos na Figura 6.

Figura 6 - Fractais aleatórios



Fonte: Autor (2021)

A Figura 6 mostra alguns exemplos de fractais encontrados na natureza. Tais fractais são gerados de forma probabilística e não faz uso de recursos computacionais assim carregando a principal característica de similaridade estatística, como exemplo no caso da árvore representada na Figura 6, a medida que é ampliado é mantido de forma aproximada do todo o fractal e mantendo a dimensão fractal.

2.3 SIMULADORES

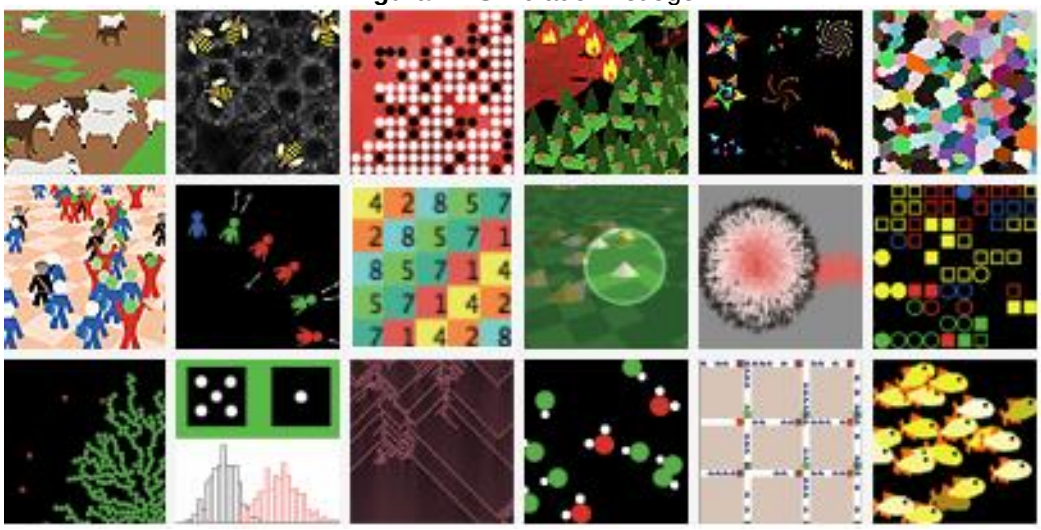
Simulação consiste em trazer a realidade para um ambiente controlado para que possa ser estudado o comportamento do mesmo, sob diversas condições, sem grandes riscos físicos e grandes custos envolvidos (TORGA, 2007, p. 54). Assim, um simulador proporciona uma maior interatividade com o usuário, e em se tratando da geometria fractal os recursos computacionais permitem uma melhor visualização do que está sendo produzido, contribuindo para uma melhor compreensão dos fractais.

A seguir serão apresentados alguns exemplos de simuladores disponíveis para a visualização e simulação de fractais.

2.3.1 NETLOGO

Netlogo foi criado por Uri Wilensky em 1999, sendo um programa de código aberto e gratuito, que pode ser utilizado em diversos sistemas operacionais tais como Mac, Windows e Linux. Com suporte a diversos idiomas. A Figura 7 mostra alguns exemplos do que pode ser gerado a partir da utilização do mesmo.

Figura 7 - Simulador Netlogo



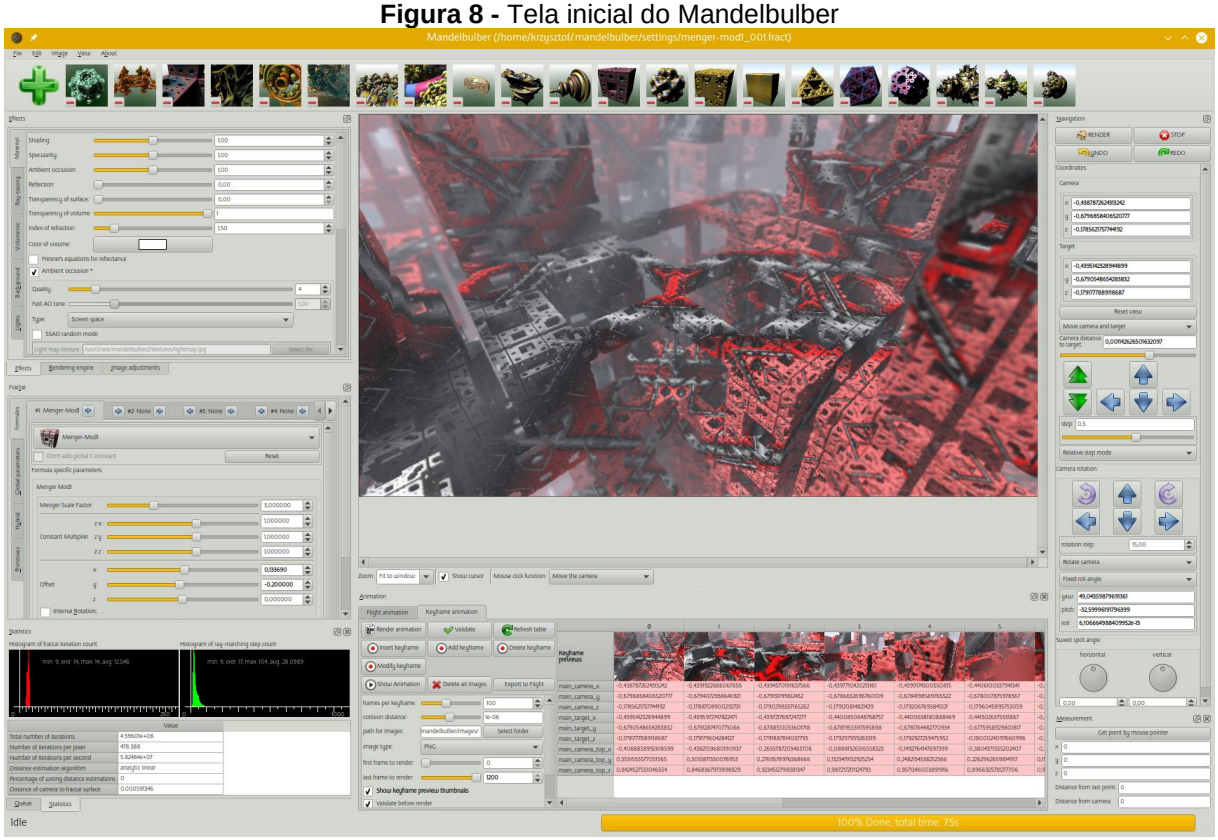
Fonte: Netlogo (2021)

O programa Netlogo (2021) “é um ambiente de modelagem programável para simular fenômenos naturais e sociais” e permite que os usuários abram simulações e interajam com elas, explorando seu comportamento (NETLOGO, 2021).

2.3.2 MANDELBULBER

Mandelbulber é um programa de simulação tridimensional baseado no conjunto de Mandelbrot, criado por Krzysztof Marczak que possui um mundo infinito de possibilidades, gerando diversas variações. Sua página inicial pode ser vista na Figura 8.

Figura 8 - Tela inicial do Mandelbulber



Fonte: Mandelbulber (2021)

Com o Mandelbulber pode-se renderizar diversos materiais personalizáveis e podendo criar imagens e vídeos do fractal escolhido (MANDELBULBER, 2021), assim, permitindo ao usuário ter acesso às suas infinitas possibilidades.

3 METODOLOGIA

Nesta seção são apresentados os materiais e métodos que foram utilizados para o desenvolvimento do simulador.

3.1 MATERIAIS

Para que fosse possível o desenvolvimento do simulador foi necessária a utilização das seguintes ferramentas:

- HTML (*HyperText Markup Language* ou Linguagem de Marcação de HiperTexto): “é o bloco de construção mais básico da web. Define o significado e a estrutura do conteúdo da web” (CONTRIBUTORS, 2021, p. 1). O HTML é uma linguagem de marcação de dados que especifica a estrutura de uma página, utilizada para compor o esqueleto de uma página web;

- CSS (*Cascading Style Sheets*): “é uma linguagem declarativa que controla a apresentação visual de páginas web em um navegador.” (CONTRIBUTORS, 2021, p. 1) responsável pela parte visual da página por dar estilo a ela por meio de linguagem declarativa a atributos (elementos) presente no HTML;

- Linguagem JavaScript: “JavaScript é uma linguagem de programação que permite a você implementar itens complexos em páginas web” (CONTRIBUTORS, 2021,p. 1). Com ela é possível controlar as tags que são elementos HTML e também gerenciar a parte lógica, responsável por realizar toda a lógica de negócios;

- P5.js: “é uma biblioteca JavaScript para codificação criativa, com foco em tornar a codificação acessível e inclusiva para artistas, designers, educadores, iniciantes e qualquer outra pessoa!”(P5, 2021). Com ela é possível gerar elementos visuais, a partir de códigos JavaScript;

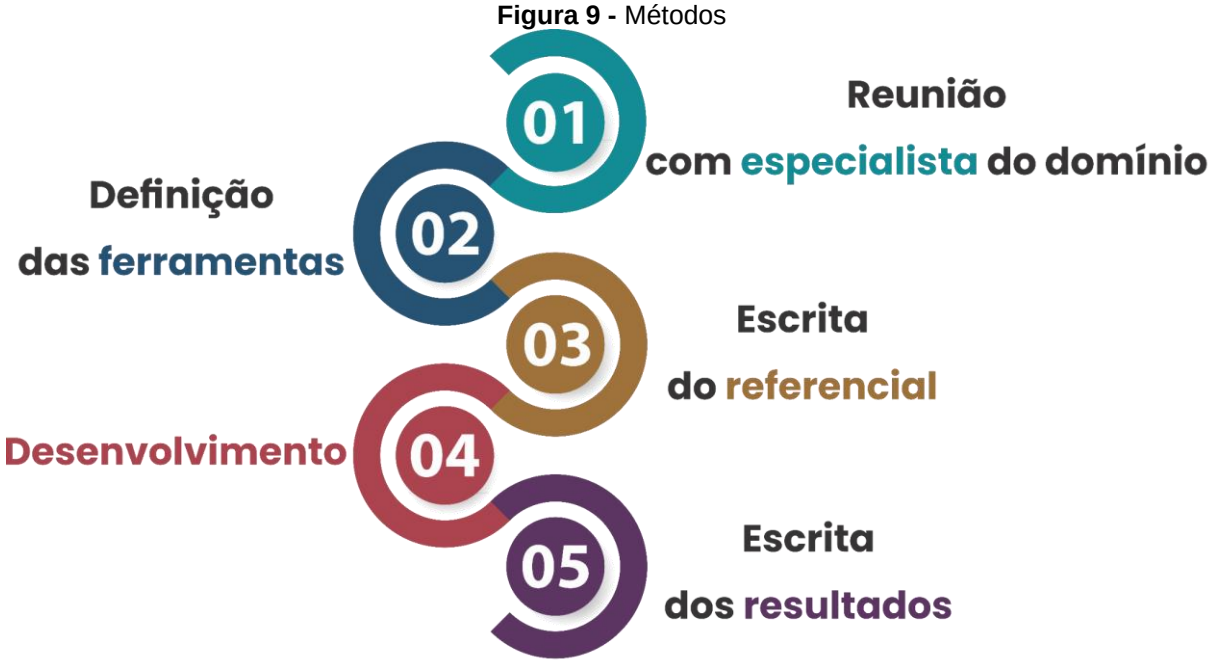
- Bootstrap : “é um framework front-end que fornece estruturas de CSS para a criação de sites e aplicações responsivas de forma rápida e simples.”(LIMA,

2021). Com a utilização desta ferramenta é possível gerar a parte visual e responsividade.

3.2 MÉTODOS

Este trabalho apresenta o desenvolvimento de uma ferramenta capaz de simular fractais de forma *online*, onde na medida que parâmetros são alterados pelo usuário altera-se em tempo real os fractais gerados. Para o desenvolvimento foi necessária a realização de algumas etapas no processo de desenvolvimento do simulador. A Figura 9 ilustra os procedimentos realizados.

Figura 9 - Métodos



Fonte: Autor (2021)

A primeira etapa consistiu na reunião com um especialista do domínio, onde foi feito o primeiro contato com a pessoa que possui conhecimento e habilidades com fractais. Nesta reunião realizou-se a seleção dos fractais que compõem o sistema, sendo realizado também um levantamento dos requisitos necessários para compor o sistema.

A segunda etapa consistiu em identificar e definir as ferramentas necessárias, para realizar o desenvolvimento do sistema que gera os fractais, tanto na parte lógica quanto na parte visual para o usuário final.

A terceira etapa consistiu no desenvolvimento do referencial teórico que utiliza fundamentos e os embasamentos sobre o trabalho, onde possuirá as informações sobre o tema do trabalho, assim tornando mais claro as informações e o intuito do trabalho que será apresentado.

A quarta etapa consistiu no desenvolvimento das funcionalidades e interfaces de visualização e manipulação dos fractais, com base nas ferramentas definidas nas etapas anteriores. Com a concepção das telas que teria o simulador, assim desenvolvendo o protótipo do simulador e por fim a validação do protótipo**.**

E, por fim, na quinta etapa da metodologia do trabalho ocorreu a escrita sobre o desenvolvimento e os resultados obtidos para conclusão do trabalho.

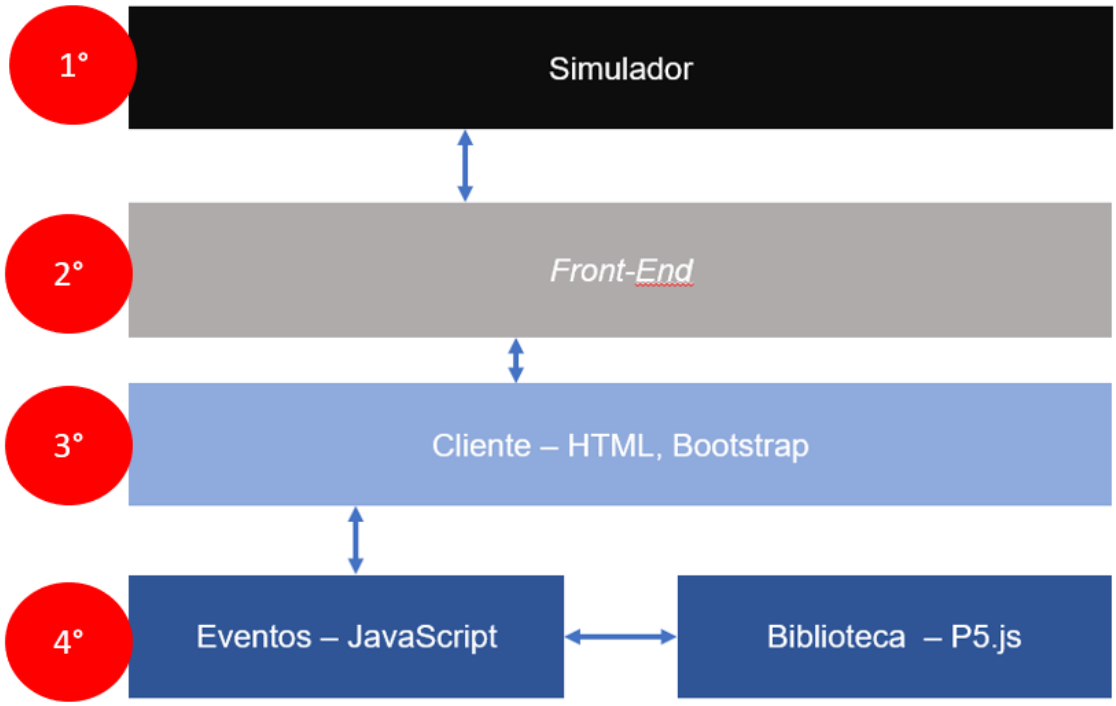
4 RESULTADOS

Esta seção apresenta o processo de desenvolvimento do simulador online de fractais, seus componentes e funcionalidades presentes, como também as interfaces de simulação: Árvore, Triângulo de Sierpinski, Conjunto de Mandelbrot e Curva de Koch.

4.1 DESENVOLVIMENTO DO SIMULADOR

O desenvolvimento do simulador foi feito em uma aplicação *Front-End* ou seja apenas versão *web.* E para compor a manipulação do simulador e também possibilitando o seu desenvolvimento foi utilizada a linguagem de programação JavaScript e a arquitetura do projeto pode ser vista na Figura 10.

Figura 10 - Arquitetura do simulador



A Figura 10 representa a arquitetura do simulador e o seu fluxo, a qual é composta por camadas. A comunicação entre as tecnologias é representada por setas bidirecionais.

O primeiro fluxo de execução da arquitetura se inicia pelo **simulador**(Figura 10 - 1) que representa a aplicação como um todo. O segundo nível ***Front-end***(Figura 10 - 2) é responsável pela comunicação entre primeira camada e a camada de **Cliente**(Figura 13 - 3) que por sua vez acopla o framework Bootstrap e o HTML para construir a interface na qual o usuário terá acesso. O quarto nível **Eventos**(Figura 10 - 4) é composto pela linguagem de programação JavaScript que é a responsável pela lógica de negócios e também implementar as funções que serão responsáveis por consumir a biblioteca P5.js.

4.2 Apresentação das interfaces do simulador

Nesta seção serão apresentadas as interfaces que compõem o simulador bem como a explicação delas. Ao acessar a página web, o usuário terá acesso à tela inicial do simulador e pode ser vista na Figura 11.

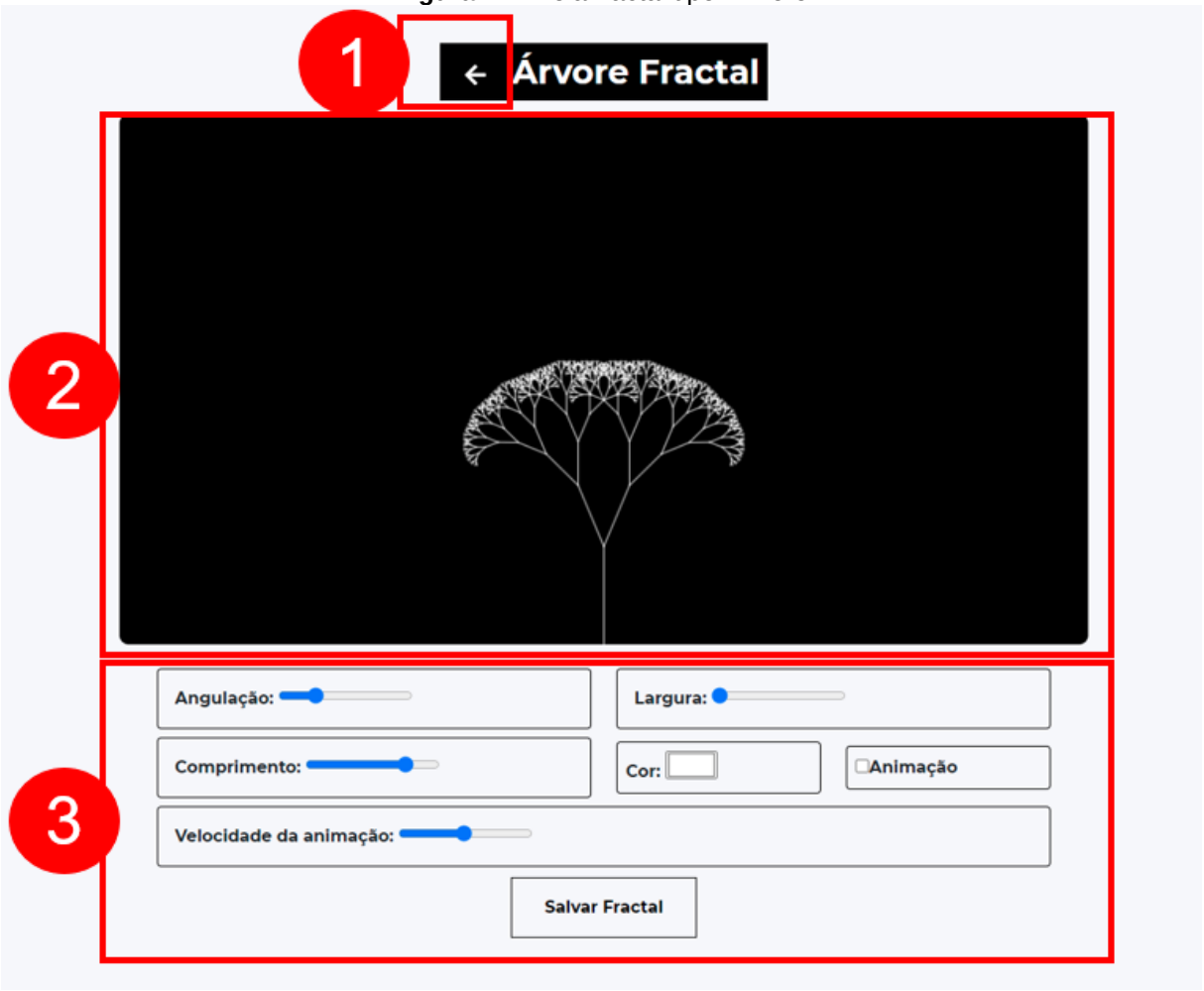
Figura 11 - Tela inicial do simulador online de fractais

Como pode ser observado na Figura 11, o simulador é capaz de disponibilizar os seguintes componentes:

- **Árvore**: módulo referente ao simulador de árvores.
- **Triângulo de Sierpinski**: módulo referente ao simulador do triângulo de Sierpinski.
- **Koch Curve**: módulo referente ao simulador Ilha de Koch.
- **Conjunto de Mandelbrot**: módulo referente ao simulador do fractal Mandelbrot.

A Figura 12 representa a tela do componente Árvore e a apresentação de sua simulação.

Figura 12 - Tela fractal tipo: Árvore



Ao analisar a Figura 12 é possível identificar as funcionalidades do simulador. Nesta tela são apresentados: o botão para voltar à tela inicial (Figura 12 - 1), o resultado da simulação do fractal tipo árvore (Figura 12 - 2). E as variáveis de controle (Figura 12 - 3) da simulação é composta por:

- **Angulação**: capaz de controlar o ângulo que cada ramo da árvore terá, e que possui valor mínimo de 1° grau e máximo de 90° graus. E tem valor pré-estabelecido de 23°
- **Largura**: tem a função de determinar a largura dos ramos da árvore, e que possui valor mínimo de 1 pixel e máximo de 4 pixels. E tem valor pré-estabelecido de 1 pixel.
- **Comprimento:** responsável por determinar o comprimento da árvore, e que possui valor mínimo de 10 pixels e máximo de 138 pixels. E tem valor pré-estabelecido de 100 pixels.

- **Cor:** responsável por determinar a cor da árvore, e que vem com cor inicial branca.
- **Animação:** responsável por animar o fractal dando a perspectiva de vento, que por padrão a opção vem desmarcada.
- **Velocidade da animação** capaz de controlar a velocidade que a animação pode ter, pode ser alterada entre o mínimo de 2 fps e máximo 60 fps. E tem valor pré-estabelecido de 30 fps.
- **Salvar Fractal** caso o usuário queira salvar uma imagem do fractal em sua máquina.

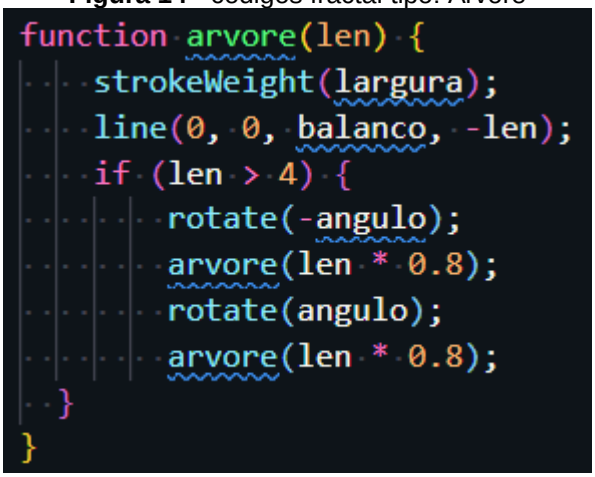
A figura 13 apresenta a criação das variáveis de controle (Figura 12 - 3). Responsáveis pela manipulação do fractal.

Figura 13 - códigos variaveis fractal tipo: Árvore

```
deslizar = createSlider(0, PI / 2, PI / 8, 0.01);
deslizar2 = createSlider(1, 4, 1, 0.01);
deslizar3 = createSlider(10, height/3.5, 100,0.00010);
deslizar4 = createSlider(2, 60, 30, 0.01);
botao = createButton("Salvar Fractal");
botao.mousePressed(salvar);
seletorCor = createColorPicker(255);
animar = createCheckbox('Animação', false);
```

Como demonstrado na figura 13 mostra a criação das variáveis de controle a partir da biblioteca p5js, onde **deslizar** corresponde a angulação, **deslizar2** corresponde a **largura, deslizar3** corresponde ao **comprimento, deslizar4** corresponde a **velocidade da animação, botao** corresponde ao botão de salvar o fractal a partir de quando é pressionado, **seletorCor** permite escolher a cor em RGB, e **animar** disponibiliza uma caixa de seleção que a partir do momento que é selecionada tem a função de animar o fractal. E a figura 14 mostra a estrutura principal do código responsável por gerar o fractal.

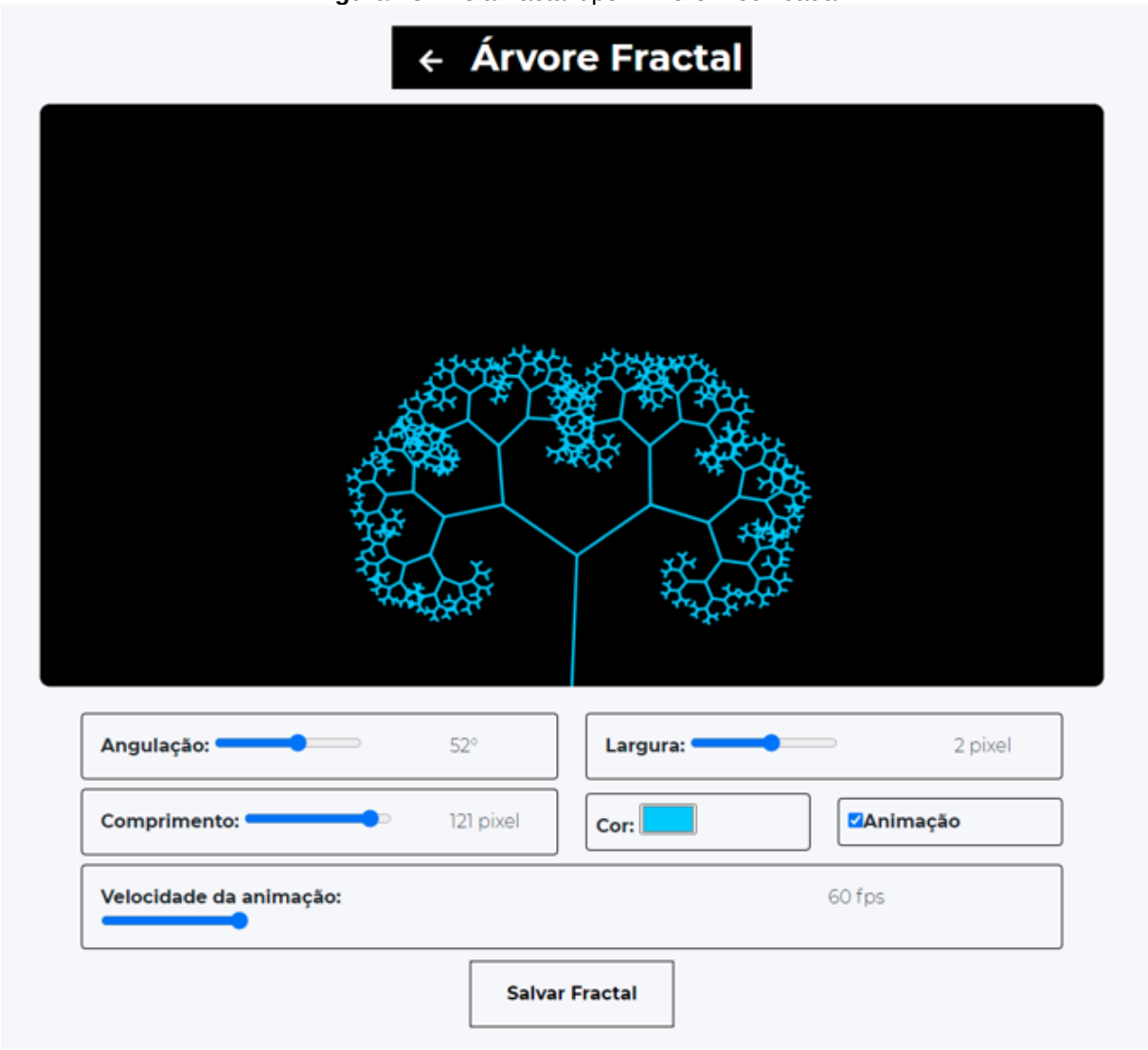
Figura 14 - códigos fractal tipo: Árvore



```
function arvore(len) {
    strokeWeight(largura);
    line(0, 0, balanco, -len);
    if (len > 4) {
        rotate(-angulo);
        arvore(len * 0.8);
        rotate(angulo);
        arvore(len * 0.8);
    }
}
```

Como demonstrado na figura 14 esta função é a responsável por por gerar o fractal, passando como parâmetro o tamanho que deseja, e a partir da manipulação das variáveis de controle(Figura 12 - 3) iterar com o fractal. A seguir na figura 15 demonstra a tela caso o usuário realize alguma modificação das variáveis de controle disponíveis.

Figura 15 - Tela fractal tipo: Árvore modificada

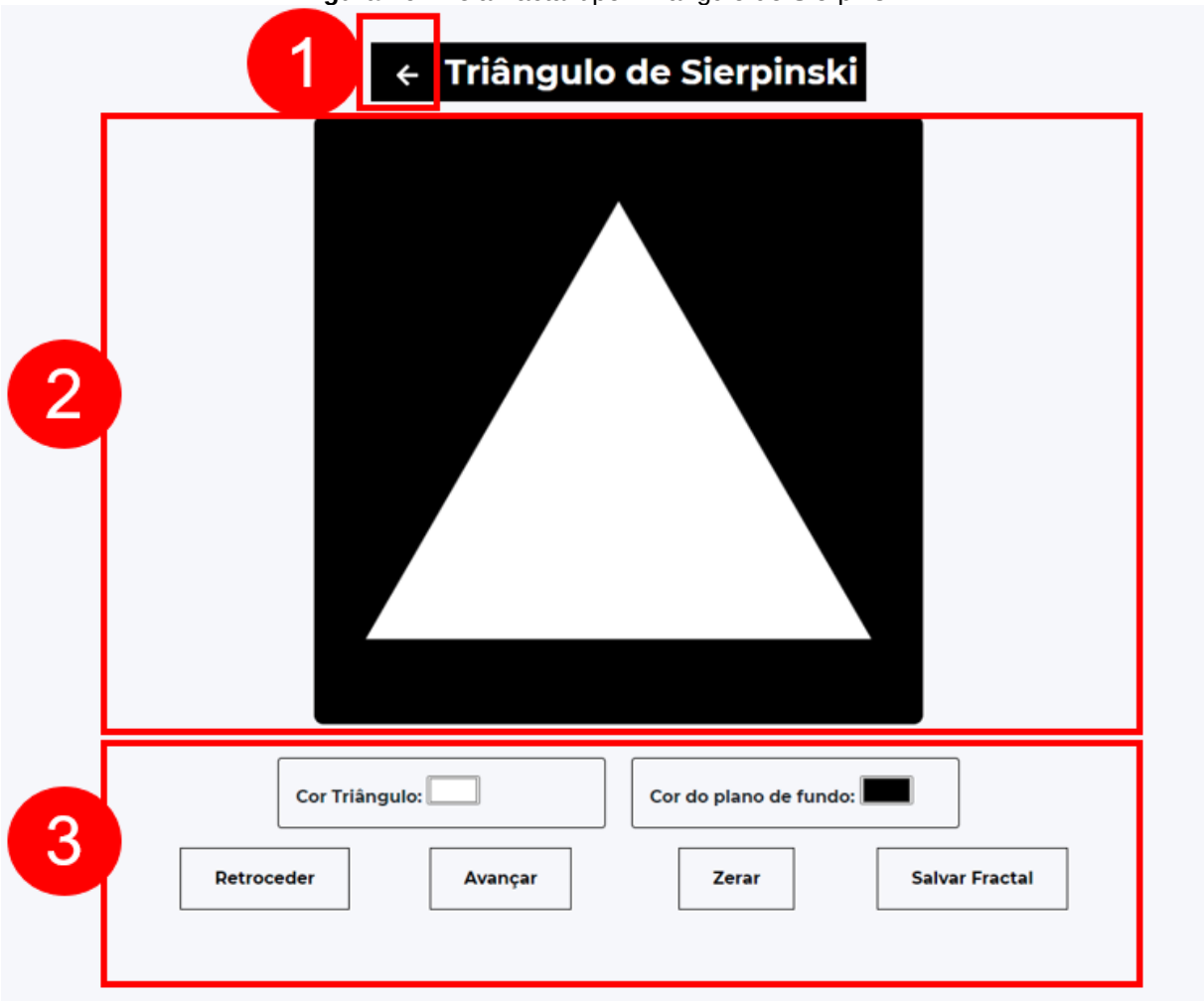


A Figura 15 representa uma entrada do usuário quando é feita a modificação dos parâmetros pré-estabelecidos informados anteriormente. Para ter chegado no que foi demonstrado, os passos seguidos foram:

1. Alterar a cor da árvore, para cor azul.
2. Alterar a angulação, para o valor de 52° graus
3. Alterar o seu comprimento, para 121 pixels
4. Alterar largura para 2 pixels,
5. Habilitar a opção de animação,
6. Alterar a velocidade da animação para 60 fps.

E a Figura 16 demonstra a tela do fractal do tipo Triângulo de Sierpinski.

Figura 16 - Tela fractal tipo: Triângulo de Sierpinski



Ao analisar a Figura 16 é possível identificar as funcionalidades do simulador. Nesta tela são apresentados: o botão para voltar à tela inicial (Figura 16 - 1), o resultado da simulação do Triângulo de Sierpinski (Figura 16 - 2). E as variáveis de controle (Figura 16 - 3) da simulação que é composta por:

- **Cor triângulo:** capaz de atribuir uma cor personalizada para o triângulo, e por padrão vem com a cor branca.
- **Cor plano de fundo:** responsável por atribuir uma cor para o plano de fundo do simulador, e que por padrão vem na cor preta.
- **Avançar:** responsável por realizar uma iteração assim gerando mais triângulos, inicialmente vem com valor 1. Podendo variar entre o mínimo de 1 e no máximo 10.
- **Retroceder:** botão responsável por retroceder a ação que foi feita, e diminuir o valor em menos 1.
- **Zerar:** funcionalidade capaz de voltar ao estado inicial do fractal. Com os valores iniciais mencionados anteriormente.

- **Salvar Fractal** caso o usuário queira salvar uma imagem do fractal em sua máquina.

A figura 17 apresenta a criação das variáveis de controle (Figura 16 - 3). Responsáveis pela manipulação do fractal .

Figura 17 - códigos variáveis fractal tipo: Triângulo de Sierpinski

```
seletorCor = createColorPicker(255);

seletorCor2 = createColorPicker(0000);

botao1 = createButton('Avançar');
botao1.mousePressed(incrementa);

botao2 = createButton('Retroceder');
botao2.mousePressed(retroceder);

botao3 = createButton('Zerar');
botao3.mousePressed(zera);

botao4 = createButton('Salvar Fractal');
botao4.mousePressed(salvar);
```

Como demonstrado na figura 17 mostra a criação das variáveis de controle a partir da biblioteca p5js, onde **seletorCor** corresponde a cor triângulo, **seletorCor2** corresponde a cor do plano de fundo**, botao1** realiza o incremento de mais triângulos, **botao2** realiza o retrocesso assim diminuindo a quantidade de triângulos**, botao3** realiza o processo de zerar assim voltando ao estado inicial, **botao4** corresponde ao botão de salvar o fractal a partir de quando é pressionado. E a figura 18 mostra a estrutura principal do código responsável por gerar o fractal.

Figura 18 - códigos fractal tipo: Triângulo de Sierpinski

```
function calculaP(x, y, w, lvl, max) {

  if (lvl == max) {
    triangle(x, y, x + w / 2, y - w * sqrt(3) / 2, x + w, y);
  } else {
    calculaP(x, y, w / 2, lvl + 1, max);
    calculaP(x + w / 2, y, w / 2, lvl + 1, max);
    calculaP(x + w / 4, y - w * sqrt(3) / 4, w / 2, lvl + 1, max);
  }
}
```

Como demonstrado na figura 18 esta função é a responsável por por gerar o fractal, passando como parâmetro x, y, w, lvl e max, tal função realiza o cálculo de três pontos, onde a partir do cálculo uma função da biblioteca p5js irá criar o triângulo no simulador(Figura 16 - 2). E a partir da manipulação das variáveis de controle(Figura 12 - 3) iterando com o fractal. A seguir na figura 15 demonstra a tela caso o usuário realize alguma modificação das variáveis de controle disponíveis.

A Figura 19 demonstra caso o usuário realize alguma modificação das variáveis de controle disponíveis.

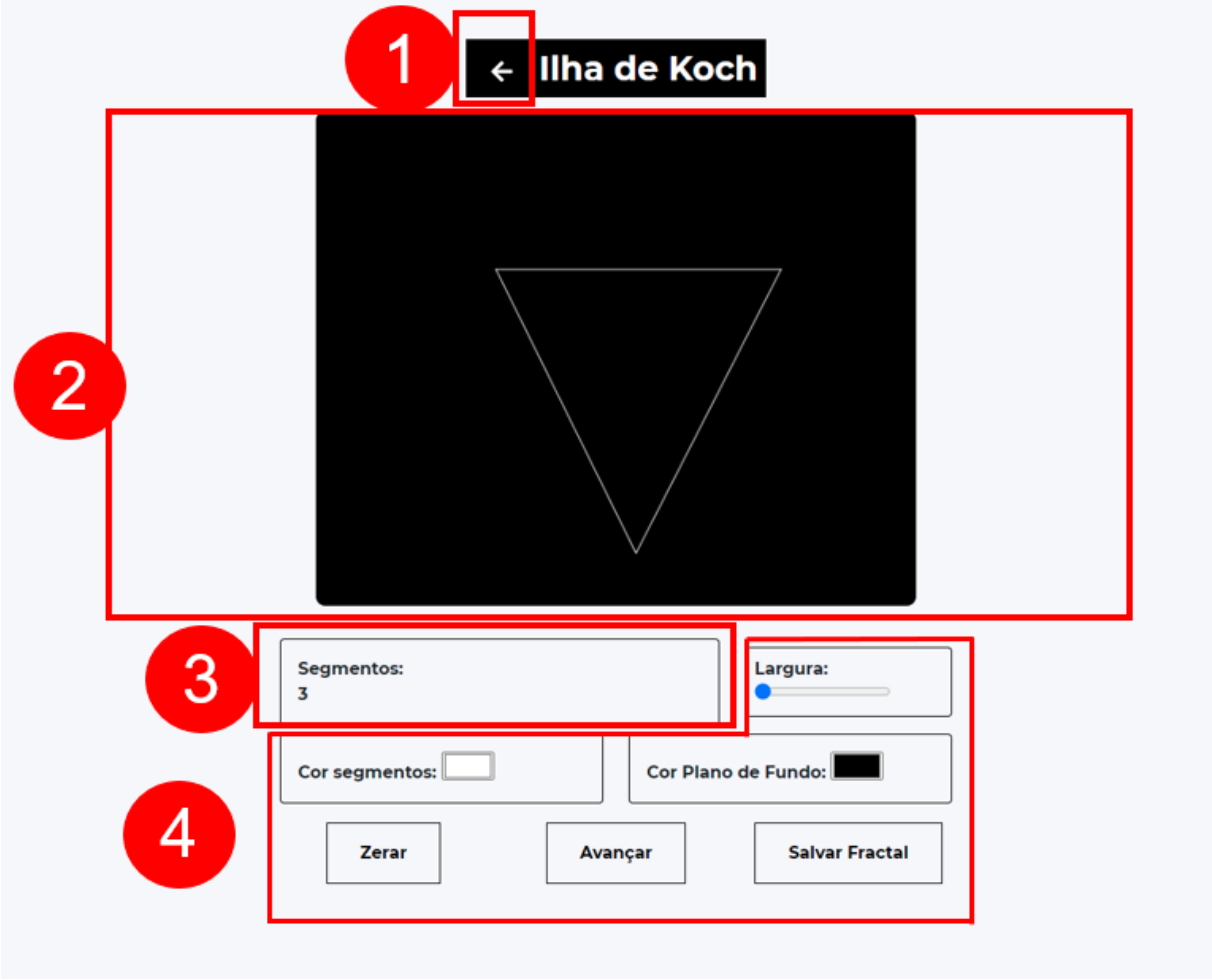
Figura 19 - Tela fractal tipo: Triângulo de Sierpinski com parâmetros modificados

A Figura 19 representa uma entrada do usuário quando é feita a modificação dos parâmetros pré-estabelecidos, para ter chegado no que foi demonstrado, o caminho foi:

1. Alterar a cor do triângulo, para a cor vermelha
2. Alterar a cor do plano de fundo, para a cor verde.
3. Avançar, realizar a ação de clique no botão 5 vezes.

A Figura 20 demonstra a tela do fractal tipo: Ilha de Koch.

Figura 20 - Tela fractal tipo: Ilha de Koch



Ao analisar a Figura 20 é possível identificar as funcionalidades do simulador. Nesta tela são apresentados: o botão para voltar à tela inicial (Figura 20 - 1), o resultado da simulação de Ilha de Koch (Figura 20 - 2) e a quantidade de segmentos gerados(Figura 20 - 3). E as variáveis de controle (Figura 20 - 4) da simulação que é composta por:

- **Largura**: tem a função de determinar a largura dos segmentos, e que possui valor mínimo de 1 pixel e máximo de 5 pixels. E tem valor pré-estabelecido de 1 pixel.
- **Cor segmentos:** capaz de atribuir uma cor única para todos os segmentos, e vem com a cor padrão branca
- **Cor plano de fundo:** responsável por atribuir uma cor para o plano de fundo do simulador, e vem com a cor padrão preta.
- **Avançar:** responsável por realizar uma iteração assim gerando mais triângulos. E atualizando a quantidade de segmentos gerados que por padrão vem pré-estabelecido o valor 3.

- **Zerar** para voltar ao estado inicial. Com os valores mencionados anteriormente.
- **Salvar Fractal** caso o usuário queira salvar uma imagem do fractal em sua máquina.

A figura 21 apresenta a criação das variáveis de controle (Figura 16 - 3). Responsáveis pela manipulação do fractal .

Figura 21 - códigos variáveis fractal tipo: Ilha de Koch

```
colorSegmento = createColorPicker(255);

colorFundo = createColorPicker(0000);

slider1 = createSlider(1, 5, 1, 0.05);

botao1 = createButton('Avançar');
botao1.mousePressed(next);

botao2 = createButton('Zerar');
botao2.mousePressed(zerar);

botao3 = createButton('Salvar Fractal');
botao3.mousePressed(salvar);

atualizaSegmentos(segments.length);
```

Como demonstrado na figura 21 mostra a criação das variáveis de controle a partir da biblioteca p5js, onde **corSegmento** corresponde geral dos segmentos, **colorFundo2** corresponde a cor do plano de fundo**, botao1** realiza o incremento degerando mais segmentos, **botao2** realiza o processo de zerar assim voltando ao estado inicial**, botao3** corresponde ao botão de salvar o fractal a partir de quando é pressionado. E a figura 22 mostra a estrutura principal do código responsável por gerar o fractal.

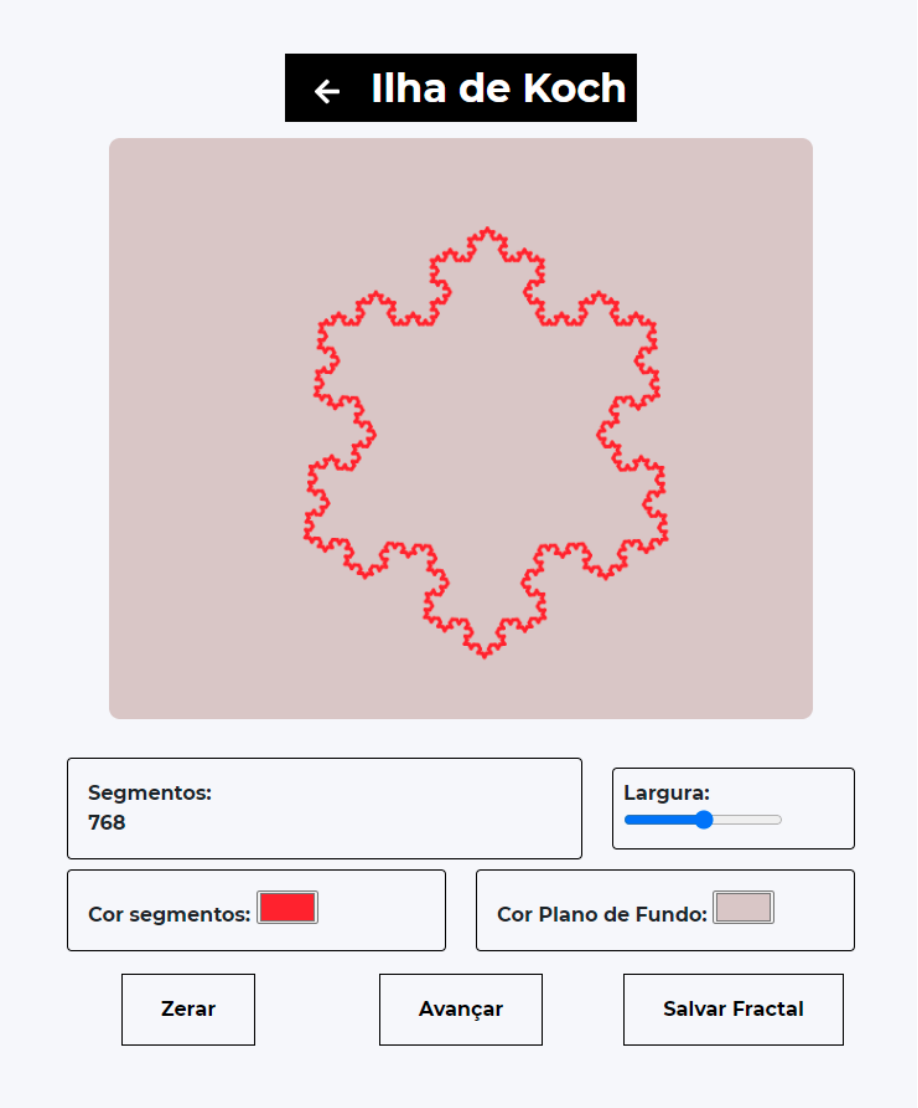
Figura 22 - códigos fractal tipo: Ilha de Koch

```
function draw() {
  background(colorBack.color());
  translate(0, 100);
  stroke(255);
  for (let s of segments) {
    show(colorSegmento.color(),slider1.value());
  }
}
```

Como demonstrado na figura 22 esta função é a responsável por gerar o fractal, através da função draw() é executada a cada segundo, a fim de atualizar de assim que houver uma alteração das variáveis de controle. A seguir na figura 23 demonstra a tela caso o usuário realize alguma modificação das variáveis de controle disponíveis.

A Figura 23 demonstra caso o usuário realize alguma modificação dos parâmetros disponíveis.

Figura 23 - Tela fractal tipo: Ilha de Koch modificada

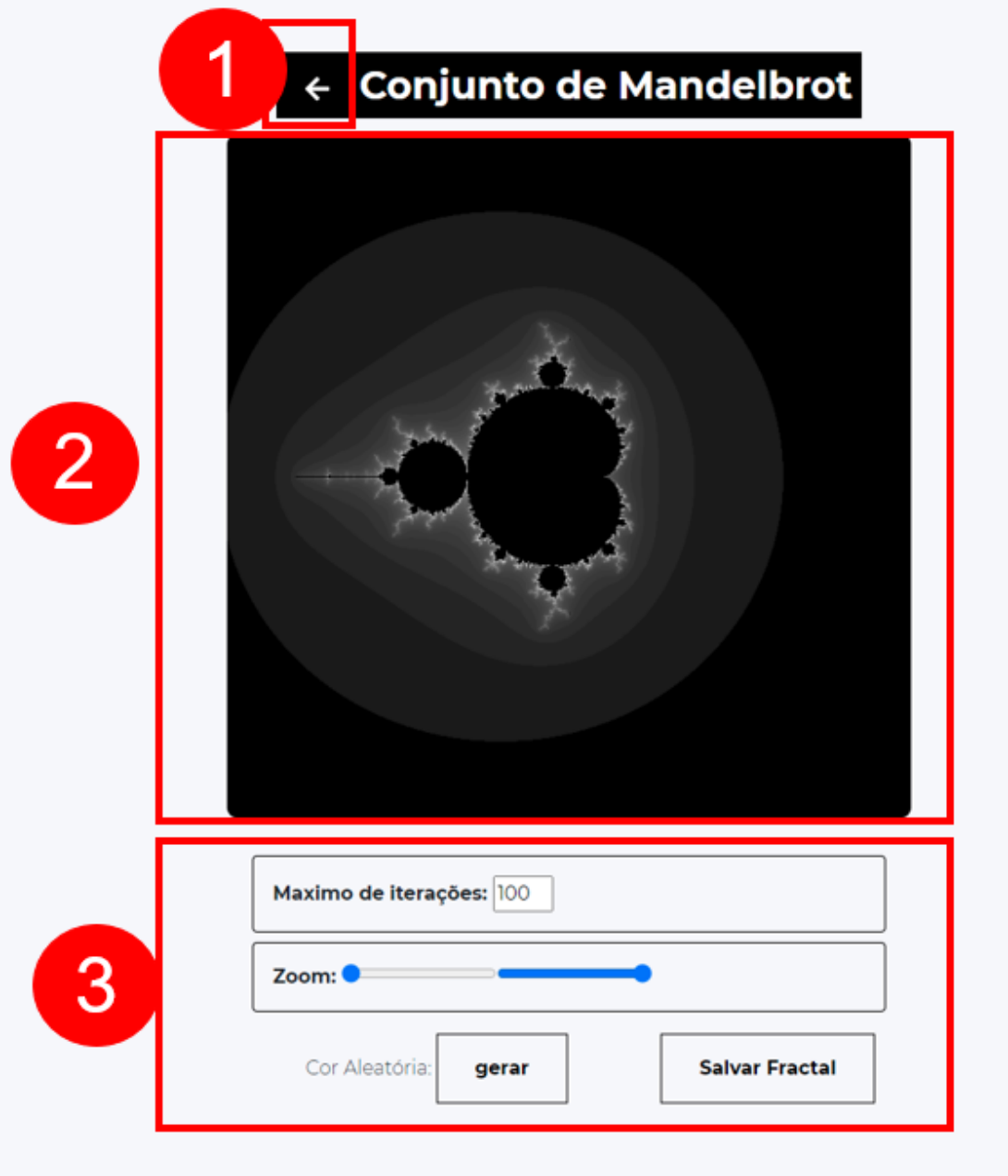


A Figura 23 representa uma entrada do usuário quando é feita a modificação dos parâmetros pré-estabelecidos, para ter chegado no resultado que foi demonstrado, o caminho foi:

1. Alterar a cor dos segmentos, para a cor vermelha.
2. Alterar a cor do Plano de fundo, para marrom
3. Realizar a ação de clique no botão **avançar** 4 vezes
4. E modificar a largura, para o valor de 3 pixels

A Figura 24 demonstra a tela do fractal tipo: Conjunto de Mandelbrot.

Figura 24 - Tela fractal tipo: Conjunto de Mandelbrot



Ao analisar a Figura 24 é possível identificar as funcionalidades do simulador. Nesta tela são apresentados: o botão para voltar à tela inicial (Figura 24 - 1), o resultado da simulação do conjunto de Mandelbrot (Figura 24 - 2). E as variáveis de controle (Figura 24 - 3) da simulação que é composta por:

- **Máximo de iterações:** responsável pela entrada numérica da quantidade de iterações que o simulador deverá efetuar para gerar o fractal, e que por padrão vem com o valor de 50, podendo variar entre o mínimo de 1 e no máximo 100.
- **Gerar:** botão responsável por gerar uma cor aleatória.

- **Salvar Fractal** caso o usuário queira salvar uma imagem do fractal em sua máquina.

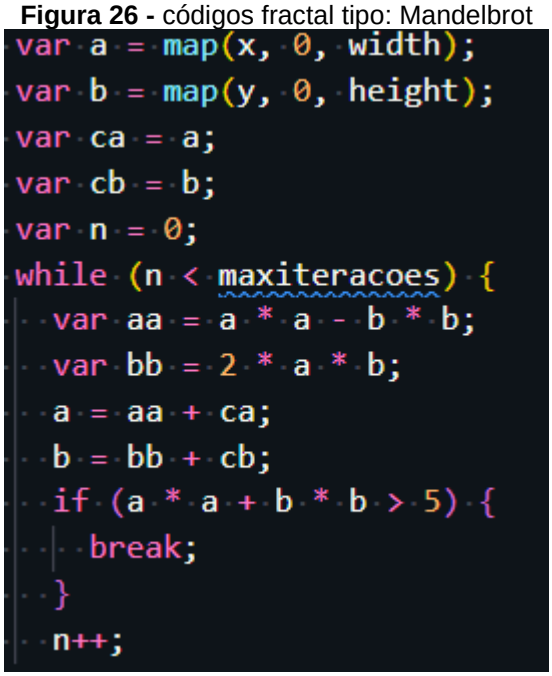
A figura 25 apresenta a criação das variáveis de controle (Figura 24 - 3). Responsáveis pela manipulação do fractal .

Figura 25 - códigos variáveis fractal tipo: Mandelbrot

```
botao1 = createButton('gerar');
botao1.mousePressed(myRandom);

botao2 = createButton('Salvar Fractal');
botao2.mousePressed(salvar);

deslizar = createSlider(-2.5, 0, -2.5, 0.01);

deslizar2 = createSlider(0, 2.5, 2.5, 0.01);

let inp = createInput('100');
inp.parent('input1');
inp.size(50);
inp.input(attMax);
```

Como demonstrado na figura 25 mostra a criação das variáveis de controle a partir da biblioteca p5js, onde **botão1** gera um número aleatório para ser utilizado na cor**, botao2** realiza o incremento degerando mais segmentos, **botao2** corresponde ao botão de salvar o fractal a partir de quando é pressionado, **deslizar e deslizar2** são referentes a manipulação do zoom. E a figura 25 mostra a estrutura principal do código responsável por gerar o fractal

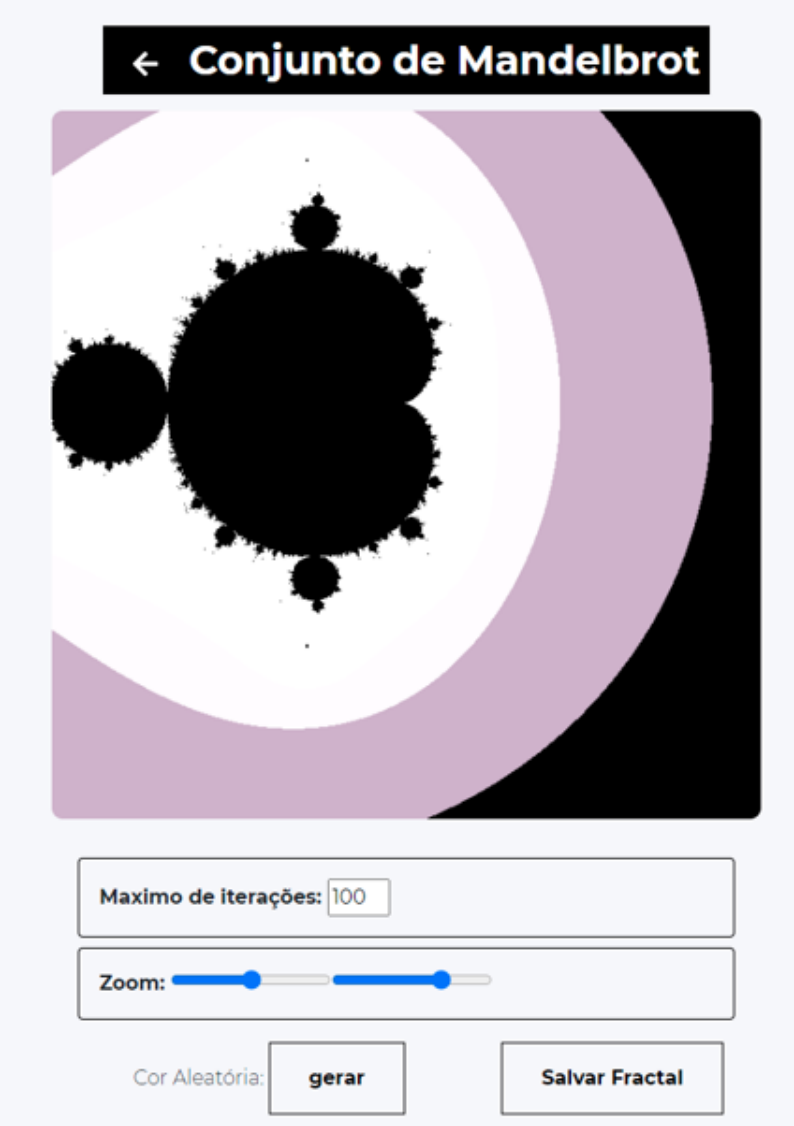
Figura 26 - códigos fractal tipo: Mandelbrot



```
var a = map(x, 0, width);
var b = map(y, 0, height);
var ca = a;
var cb = b;
var n = 0;
while (n < maxiteracoes) {
  var aa = a * a - b * b;
  var bb = 2 * a * b;
  a = aa + ca;
  b = bb + cb;
  if (a * a + b * b > 5) {
    break;
  }
  n++;
```

Como demonstrado na figura 26 esta etapa é a responsável por por gerar o fractal, gerando os pontos flutuantes com os números complexos $z = z^2 + \mathbf{c}$. Que seria calcular o número complexo **c.** A Figura 25 demonstra caso o usuário realize alguma modificação dos parâmetros disponíveis.

Figura 27 - Tela fractal tipo: Conjunto de Mandelbrot modificado



A Figura 27 representa uma entrada do usuário quando é feita a modificação dos parâmetros pré-estabelecidos, para ter chegado no resultado que foi demonstrado, o caminho feito foi:

1. Alterar o número máximo de iterações, para o valor 100.
2. Manipulado a ferramenta de zoom.
3. Gerado uma cor aleatória para o fractal.

5 CONSIDERAÇÕES FINAIS

O presente trabalho teve como objetivo central desenvolver um simulador *online* de fractais clássicos, em que o usuário poderá realizar modificações nos parâmetros assim refletindo as alterações nos fractais em tempo real. No decorrer do trabalho foi realizado reuniões com o especialista de domínio para que fosse possível determinar quais tipos de fractais iria compor o simulador.

Ao implementar o simulador, buscou-se desenvolver de forma simples e de fácil entendimento um interface para qualquer tipo de usuário. No simulador é possível realizar iterações com os fractais, assim proporcionando o acesso a um ambiente de simulação.

Assim que o simulador for implantado em um servidor e exposto a internet terá uma contribuição para a democratização do acesso, sem a necessidade de instalação de recursos adicionais.

Como trabalhos futuros, propõe-se realizar incrementos e melhorias para melhorar o desempenho do simulador, bem como adicionar novos módulos de fractais, também poderá ser implementado uma área de gerenciamento dos fractais que já foram gerados.

REFERÊNCIAS

ASSIS, Thiago Albuquerque de *et al*. Geometria fractal: propriedades e características de fractais ideais. **Revista Brasileira de Ensino de Física**, Salvador, v. 2, n. 30, p. 23041-230410, jul. 2008.

ADDISON, Paul s. **Fractals and Chaos An Illustrated Course**. Edinburgh: Institute Of Physics Publishing, 1997.

AGUILERA, Valdir. **Fractais e auto-similaridade**. 2008. Disponível em: http://www.valdiraguilera.net/fractais.html. Acesso em: 25 abr. 2021.

BARBOSA, Ruy Madsen. **Descobrindo a Geometria Fractal**: para a sala de aula. 3. ed. Belo Horizonte: Autêntica, 2005. 162 p.

CONTRIBUTORS, Mdn. **HTML: Linguagem de Marcação de Hipertexto**. 2021. Disponível em: https://developer.mozilla.org/pt-BR/docs/Web/HTML. Acesso em: 25 abr. 2021.

CONTRIBUTORS, Mdn. **CSS**. 2021. Disponível em: https://developer.mozilla.org/pt-BR/docs/Glossary/CSS. Acesso em: 25 abr. 2021.

CONTRIBUTORS, Mdn. **O que é JavaScript?** 2021. Disponível em: https://developer.mozilla.org/pt-BR/docs/Learn/JavaScript/First_steps/What_is_JavaScript. Acesso em: 25 abr. 2021.

FALCONER, Kenneth. **Fractals**: a very short introduction. Jericho: Oxford, 2013.

LIMA, Guilherme. **Bootstrap - O que é, como e quando usar?** Disponível em: https://www.alura.com.br/artigos/bootstrap?gclid=Cj0KCQiA47GNBhDrARIsAKfZ2rAFXTI45PXEBQhRSuqffi3leTsrrtA1neYSRxfu1K9B84hNAVeZ2ZIaAhm3EALw_wcB. Acesso em: 01 dez. 2021.

MANDELBROT, Benoît B.. **Fronteiras da Ciência: desenvolvimentos recentes, desafios futuros**. Coimbra: Gradiva, 2003. Disponível em: https://digitalis.uc.pt/handle/10316.2/32658. Acesso em: 25 abr. 2021.

MANDELBULBER. **Universal Idea**. Disponível em: https://github.com/buddhi1980/mandelbulber2. Acesso em: 05 jul. 2021.

NEGRI, Marília Gomes. **Introdução ao Estudo dos Fractais**. 2014. 60 f. TCC (Graduação) - Curso de Mestre em Matemática., Universidade Federal de Goiás, Goiania, 2014.

NETLOGO. **What is NetLogo?** 2021. Disponível em: https://ccl.northwestern.edu/netlogo/docs/. Acesso em: 05 jul. 2021.

RABAY, Yara Silvia Freire. **ESTUDO E APLICAÇÕES DA GEOMETRIA FRACTAL**. 2013. 89 f. Dissertação (Mestrado) - Curso de Matemática, Universidade Federal da Paraíba, João Pessoa, 2013.

ROSSETTO, Anubis Graciela de Morais. **LINGUAGEM DE PROGRAMAÇÃO WEB**. Pelotas: Universidade Aberta do Brasil do Instituto Federal Sul-Rio-Grandense, 2012. Disponível em: http://tics.ifsul.edu.br/matriz/conteudo/disciplinas/_pdf/lpw.pdf. Acesso em: 25 abr. 2021.

TORGA, Bruno Lopes Mendes. **Modelagem, Simulação e Otimização em Sistemas Puxados de Manufatura**. 2007. 139 f. Dissertação (Mestrado) - Curso de Engenharia de Produção, Universidade Federal de Itajubá, Itajubá, 2007. Disponível em: https://repositorio.unifei.edu.br/xmlui/handle/123456789/1853. Acesso em: 01 jul. 2021.