Universidade de São Paulo
Instituto de Matemática e Estatística
Bachalerado em Ciência da Computação

André Luiz Akabane Solak
Stephanie Eun Ji Bang

# eJê: construção de uma ferramenta para arte computacional

São Paulo
Dezembro de 2019

# eJê: construção de uma ferramenta para arte computacional

Monografia final da disciplina
MAC0499 – Trabalho de Formatura Supervisionado.

Supervisor: Prof. Dr. Fabio Kon
Cosupervisor: Jê Américo

São Paulo
Novembro de 2019

# Agradecimentos

Aos nossos orientadores Fabio Kon e Jê Américo, pelo suporte e paciência durante o projeto.

Aos nossos amigos, cujo apoio não se resume a apenas esse trabalho, mas a todos os momentos que compartilhamos juntos. Não sabemos como seria possível ter enfrentado a graduação sem vocês.

Às nossas famílias, que sempre nos apoiaram.

# Resumo

A Arte e suas inúmeras manifestações sempre refletiram a cultura de uma sociedade inerente ao momento histórico no qual ela estava inserida. Ao analisar o atual cenário global sob este ponto de vista, é possível notar uma intersecção cada vez maior entre as artes e a tecnologia, uma vez que o mundo vive hoje uma era dominada pelos meios técnico-científico-informacionais. Nessa gênese cultural, destaca-se o emprego crescente da computação na produção artística e, visando explorar a interação entre os dois domínios, o projeto em questão consiste no desenvolvimento de uma ferramenta computacional de auxílio ao artista plástico Jê Américo. Trata-se de uma aplicação *web* que oferece recursos personalizados para a criação de novas obras que levam a identidade estética do artista. A construção do sistema recorreu a métodos e práticas usualmente empregados nos ciclos de desenvolvimento de software, como *Sprint* e *kanban*, e envolveu o estudo e aplicação de tecnologias relacionadas à programação *web*, como *Vue.js* e *HTML Canvas*. De fácil uso e extrema utilidade, a ferramenta permitiu a Jê Américo não apenas reproduzir seu estilo artístico em obras digitais de maneira rápida e mais automatizada, mas também possibilitou a exploração de novas experiências visuais graças às funcionalidades que o eJê oferece.
**Palavras-chave:** arte, desenvolvimento web, ferramenta computacional, Jê Américo, eJê.

# Abstract

Arts and its countless manifestations have always reflected the culture of a society inherent to the historical moment it was inserted. When analysing the current global scenario from this point of view, it is possible to notice an increasing intersection between arts and technology, once the world lives today a technical-scientific-informational dominated era. In this cultural genesis, the growing employment of computing in artistic productions stands out and, aiming to explore the interaction between the two domains, the present project consists of the development of a computational tool to aid the artist Jê Américo. It is about a web application that offers custom resources for the creation of new works of art that have the artist's aesthetic identity. The system creation resorted to methods and practices usually applied in software development cycles, such as Sprints and kanban, and involved the study and application of web programming related technologies, such as Vue.js and HTML Canvas. The easy to use and extremely useful tool allowed Jê Américo not only to reproduce his artistic style in digital works in a fast and more automated way, but also to explore new visual experiences thanks to the functionalities offered by eJê.
**Keywords:** art, web development, computational tool, Jê Américo, eJê.

# Sumário

# Capítulo 1

# Introdução

Presente em inúmeras e nas mais diversas áreas do conhecimento, a tecnologia exerce hoje um papel essencial no avanço e evolução dos meios, tornando-se um sinônimo de inovação num mundo que busca incessantemente modernizar-se através do desenvolvimento técnico-científico-informacional. À frente dessa tendência está a computação, campo da ciência que causou uma verdadeira revolução não só no modo como o ser humano produz e trabalha, mas também em como ele vive e se comporta, a tal ponto que hoje em dia é difícil encontrar uma área em que a computação não seja aplicada.

Em meio à profusão de áreas que encontraram utilidade para o ramo tecnológico em questão está a Artes Plásticas. Parece estranho e destoante que tal domínio humano possa ter uma intersecção com uma ciência tão técnica e exata. No entanto, o contato entre os dois deu-se já na década de 1960, quando artistas como Vera Molnár(Rizolli, 2013) e Lillian Schwartz(Schwartz, 1995) viram nos primeiros computadores uma oportunidade de criar novas estéticas e experiências visuais, sendo pioneiros do que viria a ser chamado futuramente de arte computacional.

Naquela época, computadores ainda eram máquinas inacessíveis, grandes, caras e com um poder de processamento risível perto das atuais. Entretanto, o desenvolvimento tecnológico descomunal que sucedeu-se nas décadas seguintes expandiu ainda mais o horizonte entre arte e tecnologia e, hoje em dia, a intersecção entre as duas é enorme.

Computação gráfica, realidade virtual e aumentada, visão computacional são algumas das subáreas da computação que encontraram facilmente uma finalidade no mundos das artes plásticas. Com a sociedade inserida num contexto tecnológico tão grande, vivendo a era dos *smartphones* e dispositivos *mobile*, com acesso instantâneo à rede, e uma imensa capacidade de processamento de dados, artistas estão explorando cada vez mais os recursos que a computação e a programação podem fornecer, além das implicações que o efeito desse fenômeno está gerando.

Instigado por como as máquinas poderiam auxiliar (ou até mesmo desempenhar) o papel dos artistas, o artista plástico Jê Américo se propôs a participar do projeto desenvolvido no corrente trabalho. Formado em Arquitetura e Urbanismo pela Universidade de São Paulo

em 1994, iniciou sua carreira como artista em 2008 e, ao longo dos anos, desenvolveu uma estética e estilo próprios, com obras abstratas caracterizadas pela predominância de linhas e figuras geométricas.

Com obras de visual singular e único, Jê Américo interessou-se pela possibilidade de reproduzir o processo criativo humano através do uso das máquinas, por curiosidade filosófica acerca da natureza da interação entre humanos e sistemas. Dando sequência a um projeto iniciado no ano passado(Mendonça, 2018), em que ele e a estudante do Bacharelado em Ciência da Computação da Universidade de São Paulo, Isabella Mendonça, exploraram a questão através do uso de aprendizagem de máquina com redes neurais, a nova proposta de trabalho é uma ferramenta computacional capaz de reproduzir os elementos de sua obra a fim de auxiliá-lo na produção de novas, porém de maneira mais rápida e automatizada.

A ferramenta em questão trata-se de uma aplicação *web* que oferece recursos personalizados para a criação de obras que levam sua identidade artística. Logo, o programa dispõe de uma série de opções que reproduzem os traços e elementos característicos de suas produções, tais quais as camadas de linhas e figuras geométricas.

A realização deste projeto envolveu o conhecimento de ferramentas *web* para construção de aplicações. Deste modo, abriu-se margem para o estudo de *frameworks* em destaque, como o *Vue.js*, além de tradicionais recursos como *JavaScript, CSS* e *HTML*. Deste último, utilizou-se o elemento *Canvas*, em que também foi explorada uma grande intersecção entre programação e matemática, uma vez que a tecnologia dispõe-se de conceitos algébricos e geométricos para poder manipulá-lo.

O trabalho em questão foi dividido em 5 capítulos. O primeiro apresenta a definição do projeto, a motivação por trás de sua missão e os objetivos a serem cumpridos. O segundo descreve as metodologias e estudos aplicados no desenvolvimento da ferramenta e na organização do trabalho. O terceiro capítulo aborda as tecnologias utilizadas na construção do eJê, desde os recursos empregados diretamente na implementação do código do sistema até as plataformas de auxílio à produção e geranciamento. O quarto capítulo relata a evolução do projeto, descrevendo as mudanças de funcionalidade e *design* pelas quais a aplicação passou até a resolução da versão oficial. O quinto capítulo fornece resultados baseados nas obras geradas através da ferramenta, opiniões de Jê Américo e usabilidade do sistema. O sexto e último capítulo retoma resumidamente os tópicos mais relevantes acerca da elaboração do projeto e apresenta as considerações finais sobre o trabalho desenvolvido.

# Capítulo 2

# Metodologia de trabalho

Neste capítulo serão descritos os preparos, métodos e processos aplicados no desenvolvimento do eJê, incluindo o seu fluxo de produção e as bases de conhecimento para a implementação do sistema, sumarizados sob as seções Organização, Estudos, e Prototipação e Testes.

## 2.1 Organização

O projeto em questão foi organizado para execução em dupla. Para realizá-lo, implementou-se uma metodologia de trabalho baseada em algumas práticas e processos de *scrum*. Schwaber e Sutherland (2017) definem *scrum* como um "*framework* dentro do qual pessoas podem tratar e resolver problemas complexos e adaptativos, enquanto produtiva e criativamente entregam produtos com o mais alto valor possível". Desse método, foram empregados sobretudo os eventos de *Sprint*.

*Sprints* podem ser definidas como fases de um projeto que reúnem um conjunto de tarefas a serem cumpridas em determinado período de tempo (Schwaber e Sutherland, 2017). O trabalho a ser realizado era proposto a partir de *Sprints* de planejamento quinzenais junto aos dois orientadores deste projeto, onde havia definições de resultados de incremento, acompanhamento do progresso e demais decisões sobre a ferramenta. Em adição a esses encontros, também foram feitas reuniões entre os dois membros da equipe, onde eram realizadas Revisão e Retrospectiva da *Sprint*, além de diálogos semanais sobre andamento de tarefas e desenvolvimento da aplicação.

Uma outra prática adotada, também comum aos projetos adeptos ao *scrum*, foi a implantação de um sistema *kanban*. Tal termo significa *cartão* em japonês e concebe nome a uma metodologia criada na década de 60 pela empresa automobilística *Toyota*. Trata-se de um sistema de gestão de tarefas organizado de maneira extremamente visual para controle e fluxo de trabalho (Espinha, 2019). O aspecto *visual* diz respeito à implementação desse sistema: consiste em um quadro dividido em colunas onde cada uma possui uma lista de *cartões* referentes a tarefas específicas. No projeto em questão, a divisão era feita em "tare-

fas da semana”,”tarefas feita” e ”*bugs*”, sendo a última um espaço onde reportava-se falhas do programa que deviam ser consertadas.

## 2.2 Estudos

O desenvolvimento do projeto requereu alguns estudos e pesquisas prévias acerca de três assuntos: sistemas, linguagens e processo de criação do artista.

Definida a ideia de uma ferramenta computacional de auxílio para o Jê Américo produzir suas obras, os primeiros passos foram determinar alguns recursos básicos para a sua construção. Tendo em vista que o usuário alvo deste projeto utiliza *Windows 10* como sistema operacional, a primeira decisão foi a implementação de uma aplicação *web*, uma vez que poderia ser desenvolvida sob o ambiente de um sistema operacional *Linux*, mas ainda ser executado em outros, permitindo fácil acesso e atualização de versões.

Tomada tal decisão, a próxima questão pendente era a escolha das linguagens e, como o objetivo era desenvolvimento *web*, os meios mais óbvios a serem utilizados eram: *HTML, CSS* e *JavaScript*. Tratando-se de uma ferramenta de desenho, inspirada em *softwares* como o *Microsoft Paint*[1] e *AutoCAD*[2], foram analisadas as necessidades de sistema que o projeto demandaria e chegou-se à conclusão de que poderia sintetizar-se a uma aplicação de *front-end*, uma vez que não há primordialidade por serviços e recursos de *back-end* demasiado complexos. Nesse sentido, foi cogitado o uso do *framework Vue.js*, vantajoso por integrar *HTML, CSS* e *JavaScript* em códigos e estruturas que permitem o desenvolvimento de componentes reativos para interfaces *web* modernas, além de prover tratamento básico à manipulação e processamento de dados. Mais detalhes sobre o recurso serão descritos na próxima seção.

Por fim, o último objeto de estudo consistia no processo de criação artística de Jê Américo, fundamental para engendrar as funcionalidades da ferramenta e o modo de implementação delas. Entender como o artista produz suas obras foi um passo imprescindível pois determinou o que a aplicação disporia para a criação dos seus elementos artísticos característicos e o modo como deveria desenhá-los em uma tela de computador.

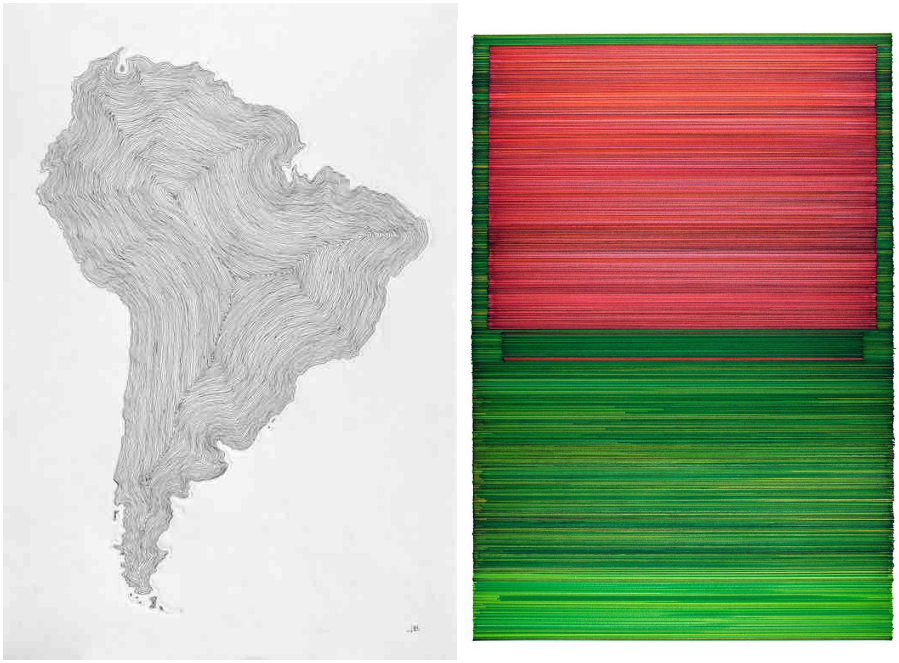
### 2.2.1 Sobre a obra de Jê Américo

Jê Américo produziu sua primeira série de obras em 2008. Nela foram retratados mapas geográficos da América do Sul através de linhas e contornos minuciosos, qualidades que futuramente viriam a tornar-se identidade estética de suas produções. Hoje, a arte de Jê é caracterizada essencialmente pela composição de linhas retas e figuras geométricas básicas.

[1]Link com análise das versões do Microsoft Paint: https://www.techtudo.com.br/listas/noticia/2017/03/microsoft-paint-todas-versoes-do-famoso-editor-de-fotos-do-windows.html. Acesso em 15 out. 2019.

[2]Link para lista de features do AutoCAD: https://www.autodesk.com.br/products/autocad/features. Acesso em 15 out. 2019.

Tais elementos foram o ponto de partida para a definição do que a ferramenta deveria provisionar: uma maneira rápida, automatizada e customizada de desenhar linhas e figuras. Deste modo, o artista tem a oportunidade de testar, experimentar e produzir profusamente, explorando novas experiências visuais.



**(a)** **(b)**

**Figura 2.1:** *Obras do artista Jê Américo com linhas e formas geométricas como seu estilo característico*

## 2.3 Prototipação e Testes

Com as tecnologias definidas e o entendimento acerca da criação das obras de Jê Américo, o passo subsequente foi a transposição da idealização da ferramenta para uma concepção material dela, isto é, a prototipação da aplicação. Segundo o livro *Design Thinking: Inovação em negócios* (2012), "o protótipo é a tangibilização de uma ideia, a passagem do abstrato para o físico de forma a representar a realidade - mesmo que simplificada - e propiciar validações". Buscando criar um modelo que representasse um aspecto geral e aberto do artefato computacional, foi elaborado um protótipo de papel.

Tal método consiste numa representação de interface gráfica sob um determinado nível de fidelidade. No caso da etapa em questão, esse nível foi de baixa fidelidade e baseou-se em uma tela feita em papel com a simulação de uma importante funcionalidade da ferramenta: a implementação de máscaras. De maneira resumida, uma máscara é um recurso que permite controlar a transparência de regiões de uma camada imagética. A ilustração 2.2 exemplifica o conceito.

Era essencial ter a ideia de máscaras previamente definida pois o recurso representaria a solução para um desafio em relação ao processo de elaboração das obras do artista paulista:

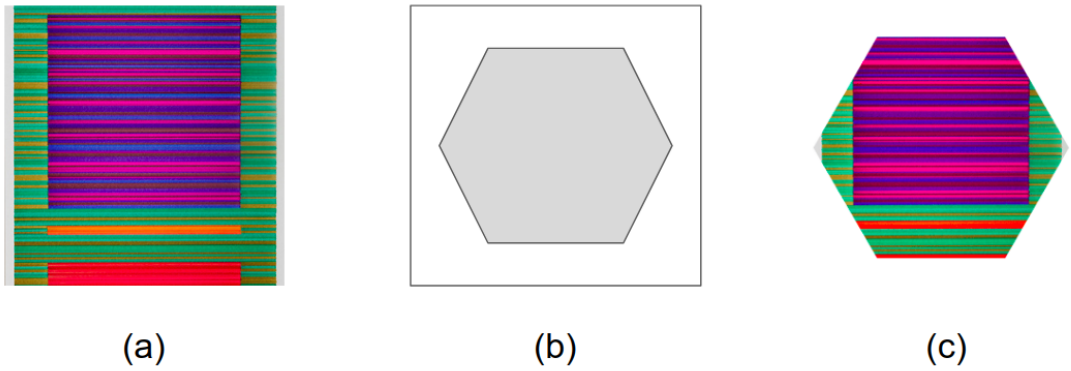
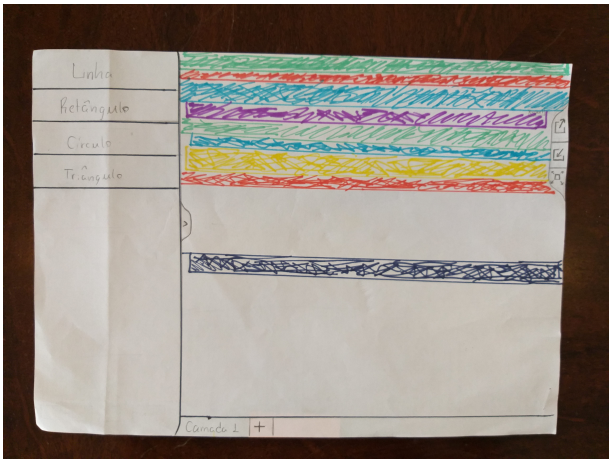



**Figura 2.2:** *A segunda imagem (b) consiste em uma máscara com uma figura hexagonal. Ao aplicá-la sobre a imagem (a), o hexágono terá 100% de transparência, ao passo que área branca omite o restante de (a), resultando na imagem (c).*

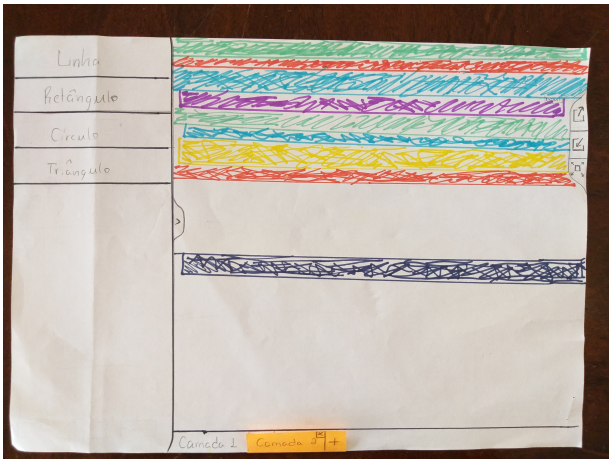
a sobreposição de figuras cujo preenchimento é feito com linhas também, característica marcante de seus quadros. Desse modo, o protótipo de papel, além de demonstrar um rascunho primitivo da interface da ferramenta, visava apresentar a execução dessa funcionalidade ao usuário e como ele poderia reproduzir elementos fundamentais de suas obras através dela. A Figura 2.3 apresenta os registros fotográficos do produto experimental.

O protótipo de papel correspondeu às expectativas em relação ao funcionamento das máscaras baseadas nas necessidades de Jê Américo, mas ainda apresentava uma interface gráfica demasiada rústica. Para definir a aparência da aplicação com um nível de fidelidade alto e mais próximo a um possível produto final, foi utilizada a plataforma *MarvelApp* para desenhar as telas do fluxo de uso. Tal ferramenta será descrita com maiores detalhes na próxima seção.

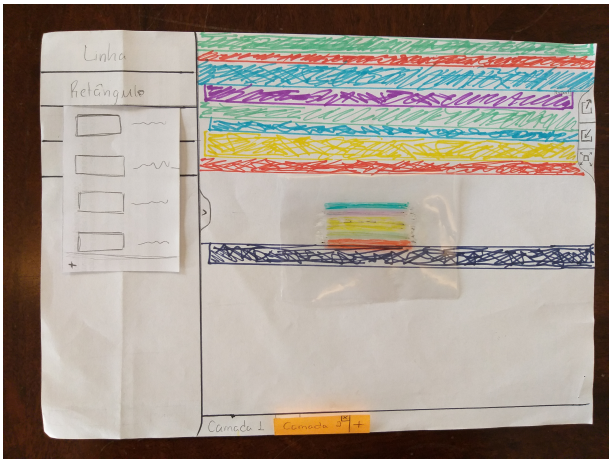
Como o processo de design do software envolveria testes com o usuário, a cada conjunto de funções novas implementadas era entregue um novo protótipo - ainda que com funcionalidade incompleta e desempenho inferior ao sistema final - para permitir o *feedback* em relação a percepções, sentimentos e avaliações, em geral, sobre as características da ferramenta. Desse modo, o sistema era adaptado de acordo com as necessidades e desejos do artista surgidos a cada nova versão. A evolução do projeto será detalhada no Capítulo 4. Vale ressaltar também que algumas dessas versões eram acompanhadas de formulários de perguntas que visavam avaliar as respectivas interface e funcionalidades.




**(a)** *primeira tela: interface da aplicação*




**(b)** *segunda tela: criação de uma nova 'layer'*




**(c)** *terceira tela: criação de uma 'mask'*

**Figura 2.3:** *Protótipo de baixa fidelidade com criação de linhas, 'layers' e 'masks'*

# Capítulo 3

# Tecnologias

Os recursos tecnológicos utilizados para o desenvolvimento deste projeto podem ser classificados de acordo com duas categorias: ferramentas de desenvolvimento e ferramentas auxiliares. A primeira engloba todos os artefatos computacionais usados diretamente no código do programa, ao passo que a segunda refere-se a aplicativos e plataformas que ofereceram suporte à organização e gerenciamento do trabalho.

## 3.1 Ferramentas de Desenvolvimento

Como já mencionado no capítulo 2 (Metodologia), as linguagens escolhidas foram *HTML, CSS* e *JavaScript*. Tratando-se de desenvolvimento *web*, as três dispensam apresentações e possuem vasta série de referências.

A seção em questão não visa aprofundar teoria e conceito sobre tais recursos, uma vez que são tecnologias extremamente difundidas e explicá-las minuciosamente seria, não apenas redundante, mas também distante do propósito deste trabalho. O foco, alternativamente, será em torno das motivações e necessidades por trás de seus usos, mostrando a aplicação que tiveram na construção do eJê.

Por essa abordagem, deve-se apenas ressaltar que, além de cumprirem com os papéis básicos na elaboração do *front-end* de uma aplicação, o *HTML* e *JavaScript* dispunham de dois artifícios poderosos para os objetivos do projeto: *HTML Canvas* e *Vue.js*. Estes, por sua vez, merecem um pouco mais atenção.

### 3.1.1 HTML Canvas

Recurso adicionado ao *HTML5*, a tecnologia do *Canvas* consiste no elemento *<canvas>*, uma *tag* usada para desenho e modelagem gráfica via código em páginas *web*. Seus casos de uso incluem desenho de figuras, composição de fotos, criação de animações e até mesmo processamento ou renderização de vídeo em tempo real. Por suas qualidades de manipulação gráfica, já tornou-se bastante conhecido entre os desenvolvedores de jogos.

Por tratar-se de um elemento *HTML*, o *Canvas* é apenas um *container* de poucos atributos que cria um plano para a geração de artes digitais, acessado através de *JavaScript*, o grande responsável por toda lógica de programação do contexto. Toda operação sobre o elemento é realizada através de um objeto denominado *context* (Fulton e Fulton, 2013), que referencia uma superfície de desenho definida por um navegador *web* que implementa o *Canvas*. Esse objeto é instanciado com parâmetros que definem algumas propriedades da figura a ser desenhada. Em termos mais práticos, a superfície é literalmente um plano cartesiano e as propriedades em questão referem-se às coordenadas e estilo.

Como cada desenho do quadro possui um contexto particular, seria necessário armazenar os valores desses parâmetros em uma estrutura que permitisse redesenhar as figuras toda vez que o *Canvas* fosse editado. Esse requisito pode ser cumprido naturalmente com os objetos *JavaScript*, sendo a opção perfeita para integrar com o resto da aplicação.

### 3.1.2 Vue.js

Criado por Even You na época que ainda trabalhava para o *Google Creative Labs, Vue.js* é um *JavaScript framework* de código aberto voltado à construção de interfaces de usuário (Galdino, 2017). Foi concebido a partir da necessidade de reutilizar códigos *HTML* a fim de evitar suas repetições, algo que consumiria muito tempo e recurso. *Vue.js* tornou-se a ferramenta perfeita para o desenvolvimento de componentes reaproveitáveis e reativos para interfaces *web* modernas - aqui entende-se por reativo a capacidade de refletir no *HTML* as alterações feitas a partir de objetos *JavaScript* que estão sendo observados.

O *framework* basicamente permite a criação de elementos que contém marcação, estilo e comportamento (*HTML, CSS* e *JavaScript*) reunidos em um só *container* de código. Assim, é possível compor estruturas de interfaces extremamente reutilizáveis denominadas *web components*. Uma das grandes qualidades do *Vue.js* é possibilitar o uso de tais estruturas como *tags* customizadas de *HTML*, facilitando a organização, a leitura e o entendimento da interface que está sendo construída.

Somado a todas as facilidades e vantagens citadas acima, um outro recurso notavelmente útil do *Vue* é o suporte à modelagem de dados através de objetos *JavaScript* simples. Em outras palavras, a ferramenta permite a construção de estruturas para armazenamento e manipulação de dados. Há inclusive um auxiliador para essa tarefa chamado *Vuex*[3]. Trata-se de um padrão de gerenciamento de estado mais biblioteca para aplicativos *Vue.js*. Ele provê um *store* (classe de armazenamento) centralizado que alimenta todos os componentes em uma aplicação de acordo com regras que garantem integridade durante a mutação de um estado, protegendo o sistema de eventuais efeitos colaterais. Dessa forma, a *store* concentra todos os dados referentes ao atual estado do programa, além das ações responsáveis pela sua mutação, centralizando todas as informações e proporcionando uma espécie de acesso compartilhado a todos os componentes. Por fim, deve-se mencionar mais um recurso oferecido pelo *Vue* de

[3]Link da biblioteca: https://vuex.vuejs.org/. Acessado em 30 out. 2019.

grande utilidade para o desenvolvimento do eJê: o *Vue CLI*, sua distribuição com interface de linhas de comando[4]. Essa ferramenta, ideal para programas mais complexos, cria uma estrutura de projeto simples, porém completa, com todas as definições de construção já configuradas e já equipada com *hot-reload* (usado para atualizar automaticamente o navegador *web* quando existe alguma alteração no código fonte do projeto) e *JavaScript lint* (usado para validar o seu código *JavaScript*, checando, por exemplo, erros de sintaxe).

### 3.1.3 Pacotes

O desenvolvimento da programa eJê envolveu a utilização de alguns pacotes com recursos auxiliares para a ferramenta. Todos eles são projetos de código aberto para o *Node.js* e instalados via NPM, um gerenciador de pacotes do *Node*.

#### prob.js

Biblioteca de funções que geram números aleatórios a partir de diferentes distribuições de probabilidade [5]. Foi empregado na geração de valores para desenhar linhas com diferentes espessuras e espaçamentos. Mais detalhes sobre a funcionalidade serão descritos no próximo capítulo.

#### vue-color

Pacote que disponibiliza a implementação de um *color picker*, componente gráfico usado para selecionar cores dos elementos dos desenhos[6].

#### vuedraggable

Componente *Vue* que implementa ação de "arrasta-e-solta"[7]. Usado para a manipulação de uma lista de objetos chamados *layers*. Mais detalhes serão descritos no próximo capítulo.

#### vue-svg-loader

Pacote que permite usar arquivos do tipo *.svg* como componentes *Vue*[8]. Usado para a inserção de todos os ícones da interface da aplicação.

## 3.2 Ferramentas Auxiliares

A construção do sistema foi conduzida com o auxílio de algumas plataformas empregadas para a organização do trabalho e produção do sistema. A seção em questão oferece uma

[4]Link para a ferramenta: https://cli.vuejs.org/. Acesso em 30. out 2019.
[5]Link para a biblioteca: https://bramp.github.io/prob.js/. Acesso em 30 out. 2019.
[6]Link para a biblioteca: https://xiaokaike.github.io/vue-color/. Acesso em 30 out. 2019.
[7]Link para a biblioteca: https://sortablejs.github.io/Vue.Draggable/. Acesso em 30 out. 2019.
[8]Link para a biblioteca: https://vue-svg-loader.js.org/. Acesso em 30 out. 2019.

descrição destas ferramentas e como elas foram utilizadas no ciclo de desenvolvimento do projeto.

### 3.2.1 GitLab

Extremamente difundido entre a comunidade de desenvolvedores e já consolidado como uma das melhores ferramentas do mercado, GitLab é uma plataforma de gerenciamento e versionamento de código baseada em *git*[9]. Além do serviço de repositórios para armazenamento de arquivos, o sistema também oferece como diferencial suporte a gestão de tarefas, CI/CD (*Continuous Integration/Continuous Delivery*) e Wiki.

O uso do GitLab foi imprescindível para a dinâmica do trabalho em dupla, uma vez que a divisão de tarefas implica a necessidade de atividade remota, algo que deve ser gerido com cuidado quando o assunto é desenvolvimento de *software*. Com a plataforma, foi possível centralizar o projeto em um local único, trabalhar em diferentes tarefas remotamente sob os paradigmas de *git* e, sobretudo, realizar revisão de código a cada *feature* adicionada ou alterada em determinada versão.

O fluxo de trabalho acontecia da seguinte forma: estabelecidas as tarefas na *Sprint* de planejamento, fazia-se a distribuição delas entre a dupla, a qual criava novas *branches* de acordo com suas respectivas atividades. Tais *branches* eram associadas a *merge requests*, que consistem em uma solicitação de verificação antes de fundir a *branch* corrente com a versão oficial do projeto, isto é, o ramo *master*. Assim, toda vez que um integrante implementava um novo recurso ou mudança de interface, o outro revisava o código, checava as alterações e aceitava a solicitação, operando o *merge* com a *master*.

### 3.2.2 Heroku

Criado em 2007 pelos desenvolvedores americanos James Lindenbaum, Adam Wiggins e Orion Henry, Heroku consiste em uma tecnologia baseada em *Cloud Computing* que oferece uma Plataforma como Serviço (PaaS), solução que abstrai detalhes de infraestrutura tipicamente associados ao desenvolver e lançar uma aplicação [10]. Deste modo, o serviço permite aos clientes concentrarem-se no desenvolvimento, execução e gerenciamento do projeto sem a complexidade do Hardware ou de servidores no processo de *deploy*, isto é, na disponibilização da aplicação aos usuários.

O serviço em questão possui integração com o *git*, o que forneceu uma ótima solução para o lançamento das versões de teste ao artista Jê Américo. Quando uma determinada fase do projeto era finalizada e o programa já oferecia novas funcionalidades ou design para serem experimentadas, realizava-se o *deploy* através do próprio *git* com uma *branch* especial associado ao Heroku, que se encarrega de executar o programa e disponibilizá-lo através de uma URL pela qual Jê Américo podia testar a respectiva versão.

[9] Link para a plataforma: https://about.gitlab.com/. Acesso em 29 out. 2019.
[10] Link para a plataforma: https://www.heroku.com/. Acesso em 29 out. 2019.

Ademais, tendo em vista a praticidade e facilidade providas pelo Heroku para a instalação, manutenção, atualização e escalabilidade da aplicação, decidiu-se que a versão final da ferramenta seria lançada pela plataforma em nuvem também.

### 3.2.3 Marvel

Marvel é um aplicativo que permite o design e criação de *wireframes*, *mockups* e protótipos de *websites* ou *apps* [11]. Além de inúmeros recursos para a edição de interfaces, é também uma ferramenta colaborativa, ou seja, pode-se criar projetos compartilhados. Foi através de tal plataforma que todos os protótipos digitais das diferentes versões do eJê foram elaborados. Uma funcionalidade muito útil do aplicativo é a possibilidade de criar fluxos de telas, isto é, uma sequência delas que simulam o funcionamento da aplicação com certo grau de fidelidade em relação à experiência de usuário. Deste modo, antes de implementar efetivamente mudanças na interface, apresentava-se o modelo criado no Marvel ao Jê Américo para o artista dar opiniões prévias sobre o design e as funcionalidades planejadas para compor um desenho.

Um outro atrativo que concebe destaque à plataforma é a precisão do design do protótipo que se pode alcançar. Como o aplicativo é específico para interfaces digitais, ele dispõe de uma série de tamanhos de telas que podem ser editadas com medidas exatas para a implementação do produto final. Dessa forma, cada elemento na tela do protótipo já possuía as medidas certas para a respectiva estilização na versão oficial da ferramenta.

### 3.2.4 Trello

Trello é um aplicativo que organiza projetos a partir de quadros, fornecendo um meio rápido e visual para verificar o que está sendo trabalhado, quem está trabalhando em quê, e em que fase se encontra o andamento de um processo [12]. Sua disposição se dá por meio de colunas preenchidas com listas de notas onde escrevem-se as tarefas da equipe. Em geral, é a ferramenta perfeita para a implantação de um sistema *kanban* em um projeto.

A adoção dessas tecnologias e da metodologia de trabalho citada na seção anterior foi um elemento chave para o processo de desenvolvimento do eJê, que requereu a criação de várias versões até se chegar na ferramenta final. A seguir, são explicadas mais a fundo as etapas de sua evolução.

[11] Link para a plataforma: https://marvelapp.com/. Acesso em 29 out. 2019.
[12] Link para a plataforma: https://trello.com/. Acesso em 29 out. 2019.

# Capítulo 4

# Evolução do Projeto

Como mencionado na Seção 2.3, a construção da ferramenta sucedeu-se de maneira gradual, com o desenvolvimento de algumas versões *beta* que eram disponibilizadas apenas para testes de usabilidade. Dessa forma, a cada variante lançada da aplicação, o artista Jê Américo retornava respostas e sugestões a fim de incrementar ou alterar as funcionalidades e interface de programa.

Este capítulo visa demonstrar a evolução do *eJê* através da síntese das edições prévias à oficial, evidenciando os critérios e condições por trás das decisões de projeto que definiram a implementação e o *design* da versão final do programa.

## 4.1 Versões *beta*

A presente seção tem como objetivo discorrer sobre as principais características e mudanças de cada versão antecedente à oficial, expondo um panorama geral acerca do desenvolvimento da ferramenta e as necessidades encontradas ao longo de sua construção. Não serão detalhados instruções de uso ou aspectos técnicos das funcionalidades apresentadas, apenas conceitos e ideias que inspiraram a implementação de componentes, funções e estruturas do código. Uma abordagem mais profunda será realizada na próxima seção 4.2, já referente a versão oficial do sistema.

### 4.1.1 Versão 1

A primeira versão do projeto, exibida na Figura 4.1, foi um protótipo extremamente rudimentar e simples, voltada mais a testes e estudos de como usar as tecnologias para o cumprimento de tarefas básicas da ferramenta do que ao desenvolvimento já visando o usuário final. Nela foi implementada apenas o recurso de desenho de linhas, sem nenhuma forma de persistência de memória - isto é, uma vez recarregada a página, o desenho era perdido -, com a opção de adicionar uma por vez ou preencher todo o espaço remanescente com linhas espaçadas uniformemente.

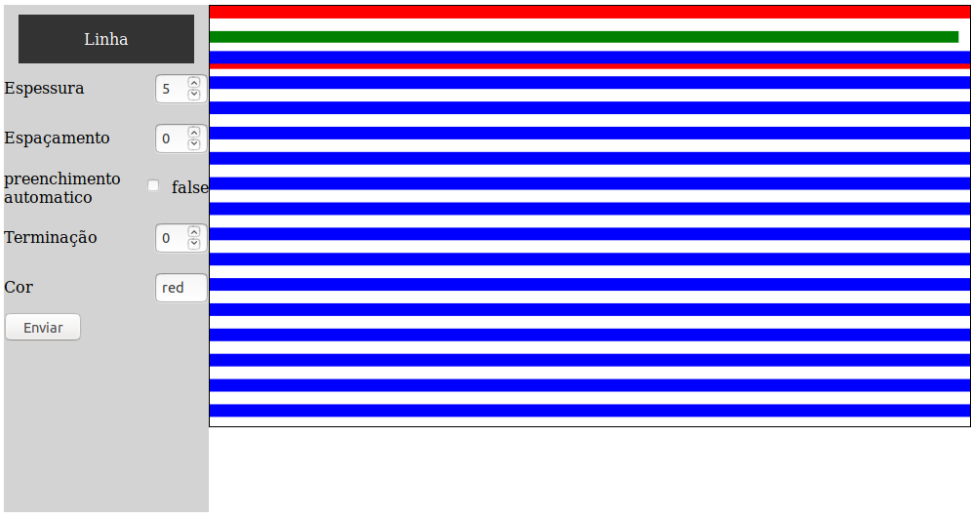



**Figura 4.1:** *Primeira versão do eJê: pode-se apenas desenhar linhas a partir da configuração dos parâmetros espessura, espaçamento, terminação e cor. Há a opção de preenchimento automático que desenha linhas uniformemente espaçadas no espaço remanescente do Canvas.*

Com nomes autoexplicativos, é fácil deduzir quais características os parâmetros `Espessura`, `Distância` e `Cor` configuram ao se desenhar uma linha. A propriedade Terminação, já menos intuitiva, é usada para definir um intervalo, entre 0 e o valor dado, em que é gerado um número aleatório correspondente a um comprimento. Esta *feature* busca reproduzir uma característica das obras de Jê Américo: a irregularidade no término do traçado das linhas, efeito que, segundo o próprio artista, é gerado naturalmente pela tinta das canetas e representa um detalhe esteticamente interessante. Sem essa particularidade na ferramenta, as obras adquirem uma aparência artificial e computadorizada.

Apesar da interface grosseira e escassez de funcionalidades, a versão em questão foi apresentada a fim de demonstrar a viabilidade do projeto através das tecnologias escolhidas e fornecer uma ideia mais tangível da aplicação, esclarecendo o que ela provisionaria e como ajudaria o artista Jê Américo na produção de suas obras.

### 4.1.2 Versão 2

A segunda versão, exibida na Figura 4.2 do projeto trouxe um considerável avanço em relação a anterior, com a implementação das principais funcionalidades previstas para desenhar linhas já em consonância com as obras de Jê Américo. As mudanças mais relevantes para se atingir o resultado foram: seleção de distribuição probabilística para o preenchimento automático das linhas (tanto para espessura quanto para espaçamento), e adição de um `color-picker` para edição das cores de linhas.

Na versão antecedente, o preenchimento automático completava o espaço remanescente do *Canvas* com linhas uniformemente espaçadas. Para proporcionar uma maior variabilidade e experimentação no processo de criação através de um fator de aleatoriedade, foram

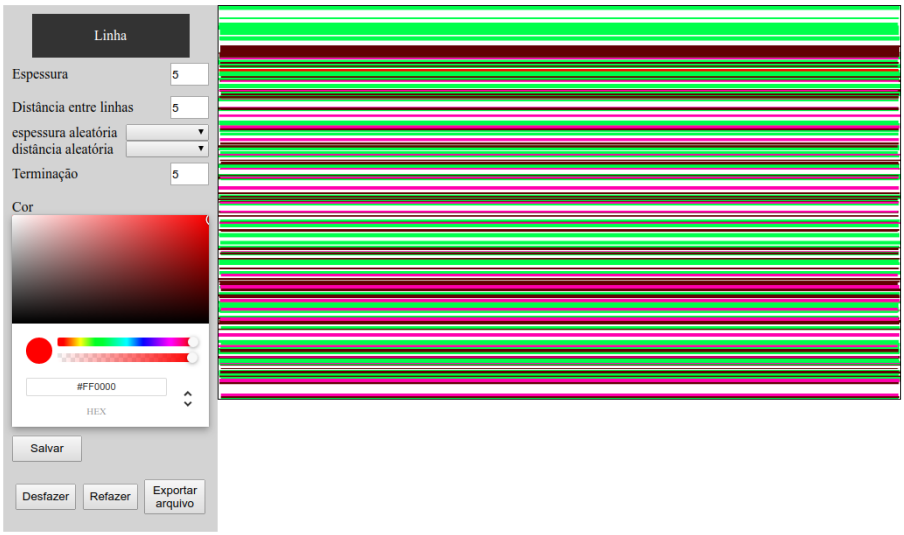



**Figura 4.2:** *Segunda versão do eJê: pode-se desenhar linhas a partir da configuração dos parâmetros espessura, distância, terminação e cor. Há a opção de preenchimento automático que desenha linhas de acordo com alguma distribuição probabilística. Ademais, foram acrescidas as funções de Desfazer, Refazer e Exportar arquivo.*

adicionadas mais quatro distribuições probabilísticas: Homogênea, Normal, Exponencial e *Poisson*. As três últimas consistem em modelos clássicos de Probabilidade, cujos parâmetros são definidos a partir dos valores dados em `Espessura` e/ou `Espaçamento`, ao passo que a primeira foi uma convenção para denominar a função que o pacote `prob.js` chama particularmente de `uniform`, mas que, segundo sua documentação, retorna um *float* uniformemente distribuído entre um intervalo definido pelos parâmetros `min` e `max`.

O `color-picker`, adquirido pelo pacote `vue-color`, fornece um componente gráfico para a escolha de cores, mas também permite a entrada de valores hexadecimais. Ademais, também foram implementados os botões de desfazer, refazer e exportar arquivo.

Mais uma mudança a ser ressaltada foi a substituição do `Espaçamento` por `Distância`. O primeiro considerava o tamanho entre o começo de uma linha até o início da subsequente, ao passo que a segunda executa o mesmo cálculo mas a partir do fim da primeira linha até o início da próxima. Tal alteração pode parecer um detalhe pouco relevante, porém implica modificações importantes na implementação do código e gera resultados diferentes ao se produzir uma obra, resultados estes pouco intuitivos para o artista sob a primeira interpretação de Espaçamento.

Por fim, uma importante estrutura foi implementada nesta fase do projeto: a *store*. Já descrita na seção 3.1.2, trata-se de uma classe, acessível por qualquer componente do sistema, de armazenamento e manipulação de dados que define o estado da aplicação e as ações para alterá-lo. No caso, este estado consiste na resolução do Canvas e nas informações das linhas renderizadas.

A Figura 4.3 apresenta algumas obras geradas por Jê Américo durante o teste de usuário da versão em questão.

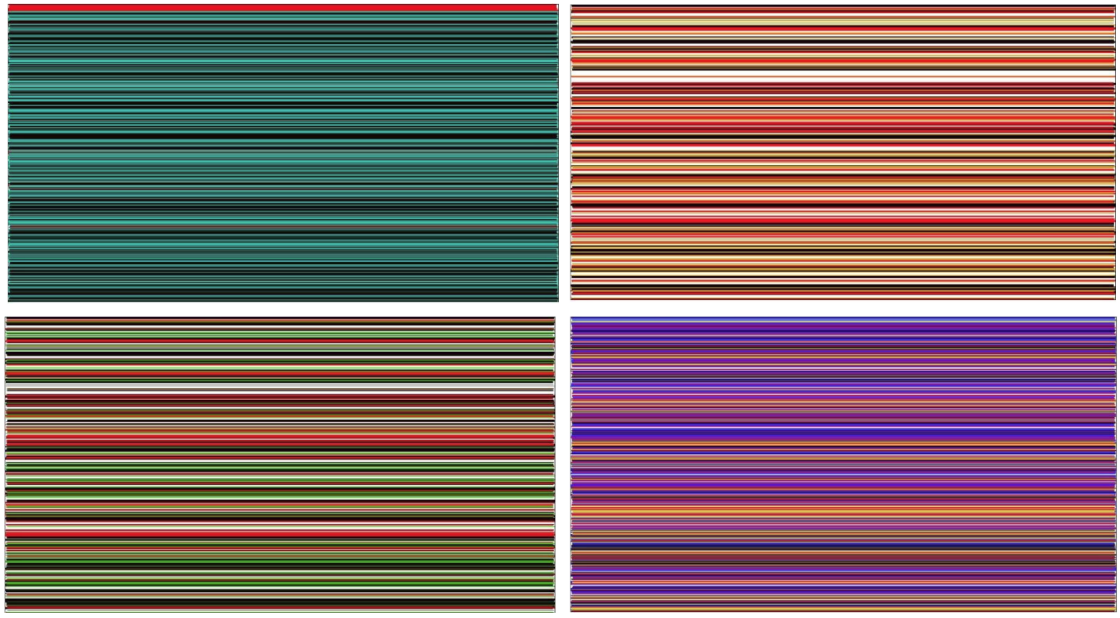
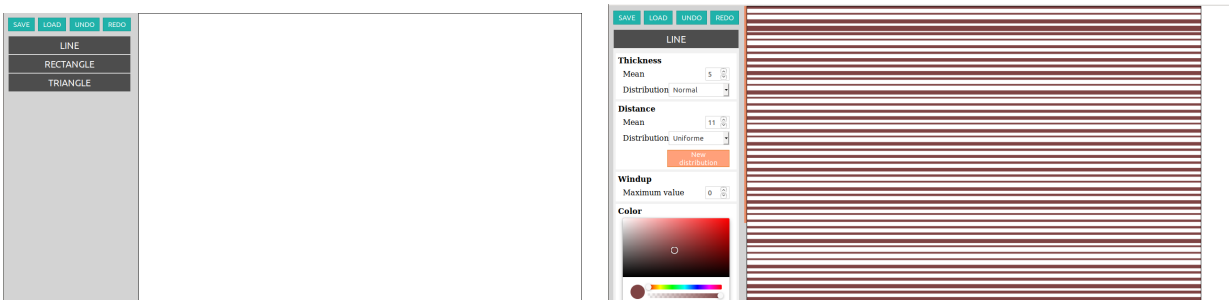


**Figura 4.3:** *Quatro exemplos de obras geradas por Jê Américo com a segunda versão da aplicação.*

### 4.1.3 Versão 3

Para a terceira versão do *eJê*, houve maior dedicação ao design da aplicação, com o redesenho da interface já visando um modelo mais próximo de um produto final. As mudanças podem ser observadas na imagem 4.4.




**(a)** **(b)**

**Figura 4.4:** *Terceira versão do eJê: as alterações em relação a versão anterior incluem a reorganização dos botões e caixas de entrada de valores, além da adição de novos elementos geométricos (neste caso, não funcionais, apenas para fins visuais do protótipo).*

A evolução da interface, agora com texto em inglês, foi acompanhado pela do código, com refatoração que buscou reestruturá-lo de maneira mais modularizada, já pensando na reutilização de componentes para a adição de futuros recursos.

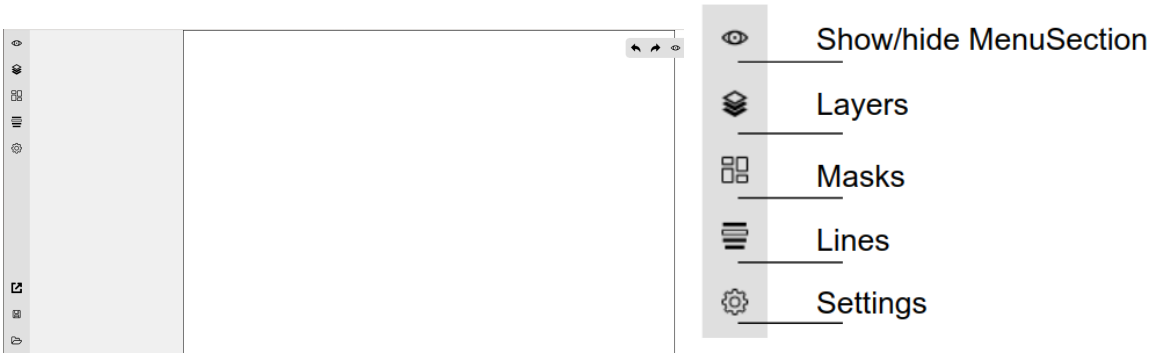
Acerca de novas funcionalidades, foram adicionados os botões *load* e *new distribution*, sendo este um gerador de nova distribuição para o modelo probabilístico selecionado. Tal opção nasceu do interesse de Jê Américo em experimentar diferentes variações visuais da mesma distribuição de maneira rápida e prática.

A versão em questão não chegou a ser lançada para teste pois houve poucos incrementos

no que diz respeito a novas *features* em relação à anterior e priorizou-se a elaboração de um protótipo já dotado de máscaras para serem testadas pelo artista.

### 4.1.4 Versão 4

O quarto modelo de sistema implementado representou um enorme passo rumo à versão oficial do projeto, com grandes mudanças em relação à interface da aplicação e ao código por trás dela. Sob o primeiro aspecto, foi elaborado um *design* completamente novo a fim de adequar melhor a ferramenta às tarefas que ela deve cumprir, além de aprimorar sua aparência com um visual mais elegante. Acerca do código, houve também uma grande reestruturação para organizá-lo de maneira otimizada às novas estruturas implementadas. As figuras 4.5 e 4.6 exibem a nova interface e os containers que a compõem, respectivamente.




**(a)** Design *da interface da versão 4* **(b)** *Listagem dos compomentes da* ToolBar

**Figura 4.5:** *Interface e novas funcionalidades da versão 4. Entre as novas funções destacam-se: opção de* show/hide MenuSection*, usado para a visualização do* Canvas *sem a interferência do referido container, ao clicá-lo,* MenuSection *é omitido e obtém-se uma visão maior e mais centrada do desenho; a criação de camadas através do componente* Layers*; a geração de máscaras por meio do componente* Masks*; desenho de linhas através de* Lines*; e finalmente edição da resolução do* Canvas *pelo componente* Settings*.*

As mudanças mais profundas foram decorrentes da introdução de dois conceitos: *layers* e *masks*. As variantes prévias do sistema foram desenvolvidas sobre uma estrutura de dados onde linhas e diferentes formatos de máscaras, como triângulos e círculos, seriam elementos primitivos, isto é, componentes independentes na constituição do desenho. Logo, a composição de uma obra consistiria simplesmente num conjunto de tais elementos, dispostos de maneira arbitrária pelo usuário. Já na versão em questão, houve uma grande alteração neste paradigma de composição através das *layers* e *masks*. Agora, uma obra não consistiria mais numa simples reunião de objetos independentes, mas sim numa sequência de camadas (as *layers*) que comportam máscaras (as *masks*, estas já descritas em 2.3). Para exemplificar melhor, observe a Figura 4.7

A principal diferença entre a decomposição I e II é a presença das *layers* que, conforme o nome sugere, trata-se de abstrações de camadas do desenho. Analisando-se o trabalho de Jê Américo, foi notado que a maioria de suas obras eram compostas por um fundo

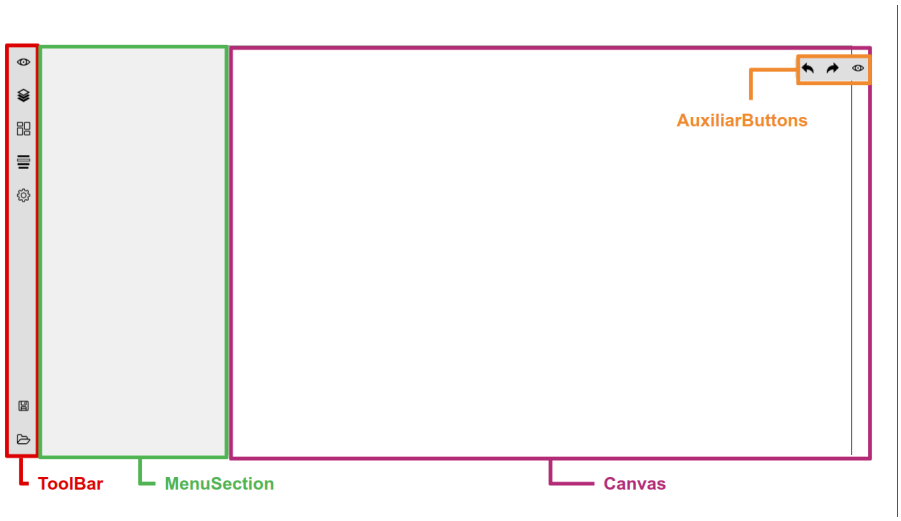



**Figura 4.6:** *A nova interface proposta dividiu a aplicação em 4 grandes containers: ToolBar, MenuSection, Canvas e AuxiliarButtons. A primeira trata-se de uma barra de seleção de tarefas, onde cada ícone corresponde a uma determinada função. A segunda, MenuSection, representa a área de configuração das estruturas que o usuário irá editar, cada qual relacionada a sua respectiva função e ícone da ToolBar. Canvas corresponde ao plano, à superfície de desenho e, por último, há AuxiliarButtons, que contém os botões de Undo, Redo e Show/Hide, este último usado para esconder o container.*

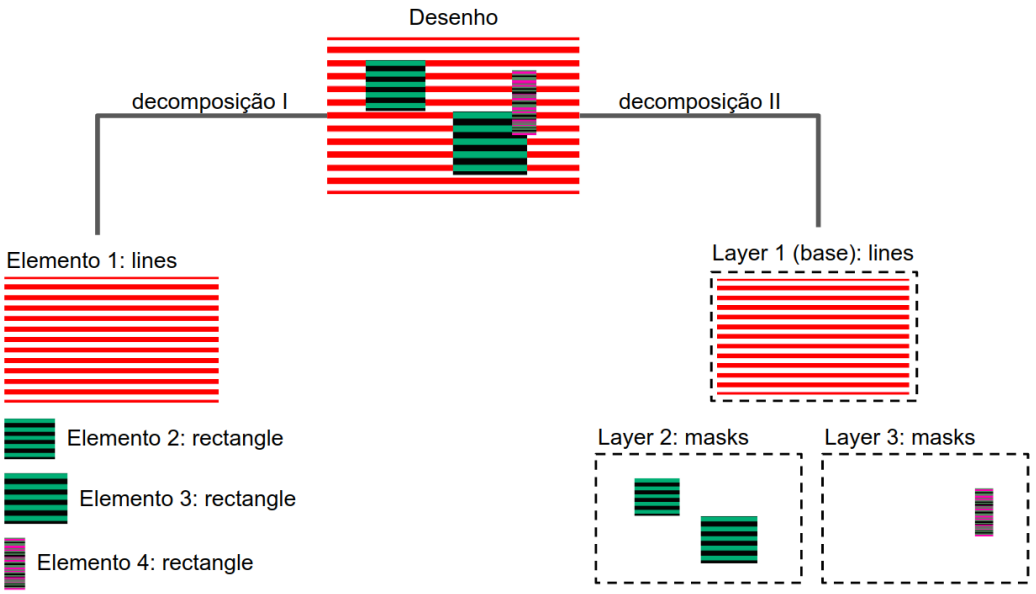
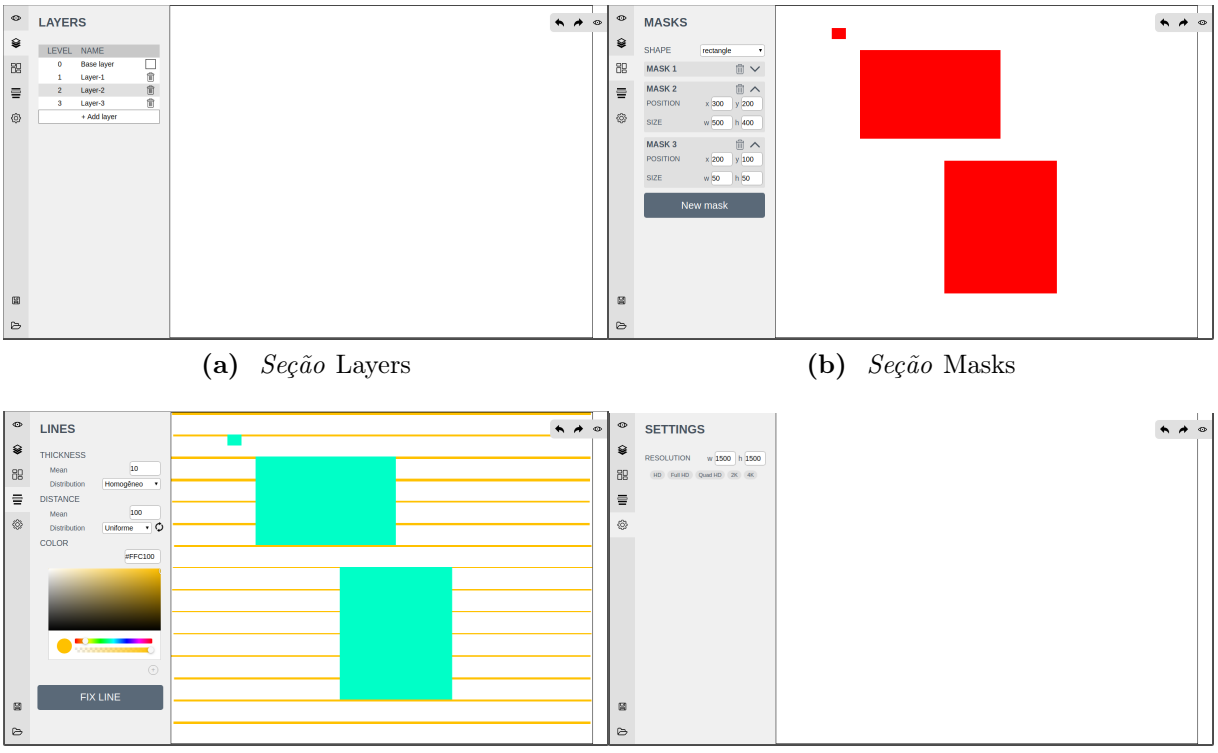



**Figura 4.7:** *Decomposição de uma imagem gerada em objetos.*

preenchido por uma determinada configuração de linhas, sobposto a um segundo plano onde há as características formas geométricas, também concebidas em linhas. Tendo em vista esta abordagem, foi implementada a classe de componentes `Layers` como um meio de modularizar os elementos do *Canvas*, tornando sua organização mais clara, pragmática e próxima ao método de produção do artista.

Além destas qualidades, `Layers` também torna mais intuitiva a ideia das máscaras, representadas no código pela nova classe `Masks`. Na camada 2, por exemplo, há duas `masks` retangulares. Segundo o próprio conceito de máscaras, estes componentes representam um "recorte" visual do fundo sobre o qual estão aplicadas, ou seja, a `Layer 2` é que está preenchida por linhas verdes e pretas, mas o que se vê são as áreas delimitadas pelas *masks*. Pelo paradigma demonstrado na decomposição I, presente nas versões anteriores, as duas figuras em questão são interpretadas como elementos individuais, cada qual com sua própria configuração de linhas.




**(a)** *Seção* Layers **(b)** *Seção* Masks

**(c)** *Seção* Lines **(d)** *Seção* Settings

**Figura 4.8:** *Visão geral das seções do sistema.*

É importante ressaltar uma convenção adotada acerca da relação entre `Layers` e `Masks` nesta versão: a primeira camada, denominada `base`, seria fixa e proibida de conter máscaras. Tal padrão foi empregado como uma forma de preservar a ideia de um fundo único composto somente por linhas, ao passo que as camadas conseguintes seriam exclusivamente as portadoras de máscaras.

Em adição aos dois conceitos introduzidos nesta estapa do projeto, mais algumas ideias foram implementadas. Uma delas foi a opção de aplicação de uma cor sobre o fundo do

*Canvas* (camada `base`), função sugerida pelo próprio Jê Américo, que desejava preencher os espaços em branco remanescentes entre as linhas.

Outro recurso incorporado à ferramenta foi a escolha da resolução do *Canvas*, função essencial ao artista caso haja o interesse ou necessidade de produzir versões físicas das obras através de impressões - o que implica a escolha de dimensões do papel - ou até mesmo para exibição digital, mas em proporções aumentadas.

Por fim, uma última funcionalidade acrescentada foi um componente de paleta de cores, recurso incluído junto ao `color-picker` que constitui um elemento gráfico para salvar as cores que sejam do interesse do artista deixar a sua disposição. Uma visão geral de todos componentes funcionais da versão pode ser observada na Figura 4.8.

Resultados gerados pelo artista com a versão em questão da ferramenta foram reunidos na imagem 4.9

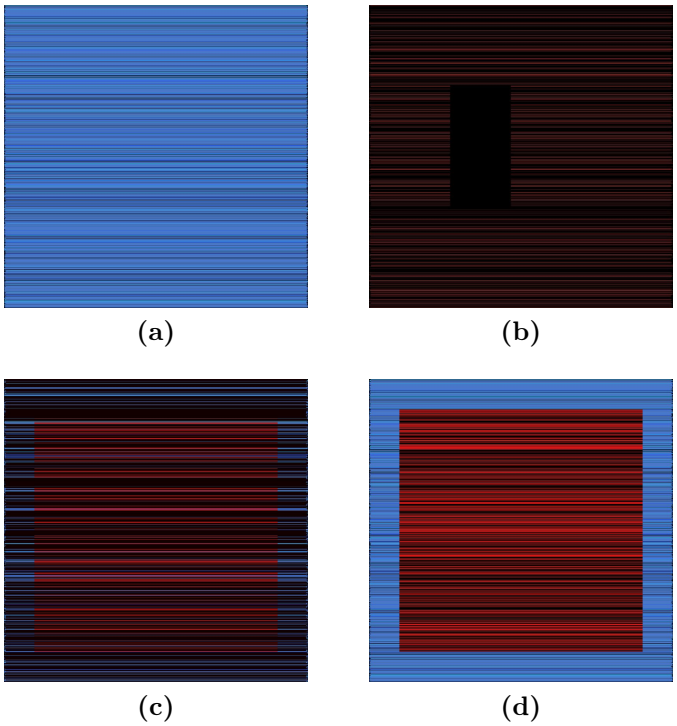



**Figura 4.9:** *Obras geradas por Jê Américo com a versão 4 da ferramenta.*

## 4.2 Versão Oficial

A última e vigente versão manteve-se fiel ao *design* de seu antecessor, sem mudanças muito profundas na interface e com novas funcionalidades adicionadas. Nesta seção, a aplicação será abordada sob dois aspectos: seu funcionamento em detalhes, isto é, como os conceitos apresentados até o momento foram traduzidos para funcionalidades e como o usuário a utiliza para a confecção de uma obra; e a implementação do código, a saber, a arquitetura das estruturas de armazenamento e manipulação de dados.

### Funcionaliades e uso da aplicação

Os recursos usados para a criação do desenho estão associados a um determinado componente que pode ser acessado pelos seus respectivos ícones na ToolBar. Na versão oficial, existem 6, exibidos na Figura 4.10.

Em relação à versão anterior, *Show/hide* `MenuSection`, `Layers` e `Masks` mantiveram as mesmas funções, ao passo que `Lines` e `Settings` tiveram seus ícones removidos da `ToolBar` e suas seções realocadas em outros componentes. Os três ícones inferioes correspondem, respectivamente, às funções `Save File`, `Open File` e `Export File`. O primeiro salva, na máquina do usuário, o arquivo *.json* responsável pelo armazenamento das informações necessárias para a recuperação do quadro; o segundo abre o mesmo tipo de arquivo para redesenhar o estado do *Canvas*; já o terceiro e último oferece duas opções de exportação de imagem para o formato `.png`: uma que renderiza o desenho corrente do *Canvas* e outra que produz e exporta uma variação aleatória na distribuição das linhas do mesmo. Esta função foi adicionada como um recurso de auxílio ao artista para gerar *frames* que podem ser utilizados na produção de animações, através de alguma ferramenta externa de edição de vídeos, por exemplo. Deste modo, Jê Américo pode produzir desenhos com pequenas alterações que, exibidos sequencialmente num determinado intervalo de tempo, atribuem movimento o obra.

A primeira alteração importante em relação ao *design* da interface é a presença do formulário `Lines` tanto em `Layers` quanto em `Masks`. Na versão antecessora, `Lines` era uma seção separada e compartilhada, usada para o preenchimento tanto da camada base quanto das máscaras. Visando atender à sugestão de Jê Américo, que achava inconveniente para sua experiência de usuário ter que mudar de seção toda vez que tinha que editar ou adicionar linhas, o componente foi fixado na metade inferior da `MenuSection` quando necessário.

Sobre a configuração das linhas, os parâmetros continuaram os mesmo das versões anteriores, com exceção do `Windup/Terminação`, cuja a entrada foi removida e os valores, delegados à geração automática na própria função do código, uma vez que não era do interesse do artista manipular esta propriedade.

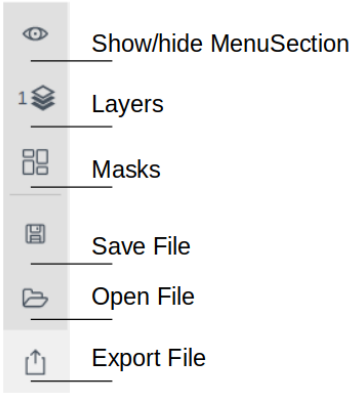
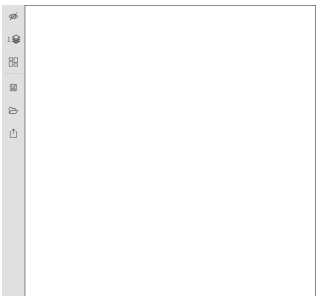
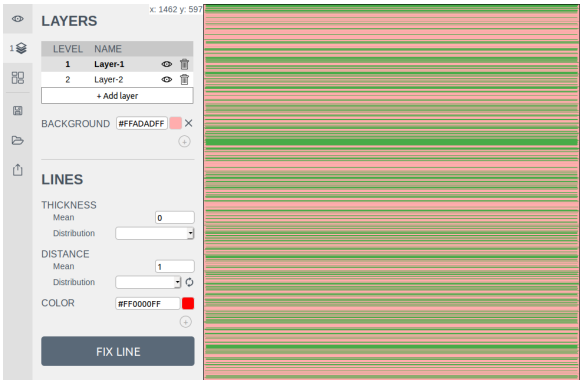



**Figura 4.10:** *Listagem dos ícones disponíveis na ToolBar*

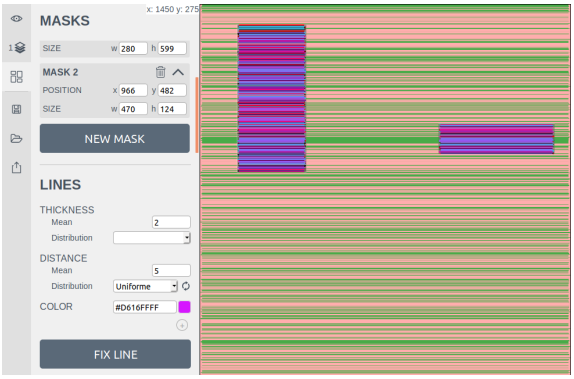

**(a)** *Tela inicial do sistema: Settings da resolução do Canvas*



**(b)** *Tela com MenuSection e AuxiliarButtons escondidos*




**(c)** *Tela com seção Layers selecionada e desenhada*




**(d)** *Tela com seção Masks selecionada e desenhada*

**Figura 4.11:** *Visão geral das seções do sistema.*

Como exibido na tela 4.11.c, através da seção `Layers`, o usuário pode adicionar camadas clicando em "+ Add layer", remover clicando no ícone de lata de lixo e, sobretudo, mudar as camadas de posição através da ação arrasta-e-solta. Este recurso implica a mudança da ordem das camadas no desenho também, exercendo funções de *mover-para-trás* e *trazer-para-frente*, conforme demonstrado pela imagem 4.12. Por fim, manteve-se a opção de preencher os espaços em branco remanescentes entre as linhas através do parâmetro `BACKGROUND`, que pinta o fundo inteiro (inicialmente transparente) do *Canvas* com uma cor selecionada por meio do `color-picker` ou valor hexadecimal.

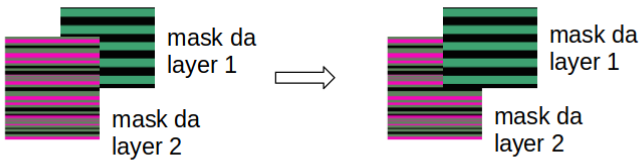



**Figura 4.12:** *Alterar posição das camadas implica mudança na ordem em que as máscaras e linhas são desenhadas*

Dando sequência à `Layers`, encontra-se a seção `Masks`, área de edição das máscaras. Toda *mask* está associada a uma *layer*. Logo, para trabalhar com as máscaras de uma determinada camada, esta deve ser selecionada na lista de `Layers`. A ferramenta permite

apenas a geração de máscaras retangulares e há duas maneiras de criá-las: clicando-se no botão *NEW MASK* e configurando os parâmetros `POSITION` e `SIZE`, ou desenhando manualmente com o próprio *mouse* (para o auxílio da tarefa há um indicador de coordenadas do *Canvas* sincronizado com o cursor do mouse). É possível removê-las clicando no ícone de lata de lixo.

Aqui deve-se mencionar uma mudança importante acerca das *layers* e *masks* em relação a versão anterior. Antes, o *Canvas* possuía a camada base, onde era proibida criação de máscaras, que por sua vez só podiam ser definidas em camadas conseguintes. Na versão oficial, tal paradigma foi preteriado e a camada `base`, removida. Agora, qualquer camada pode ser constituída tanto de máscaras quanto linhas de fundo. Esta alteração decorreu da vontade de Jê Américo de possuir maior liberdade para manipular a disposição dos conjuntos de linhas através das ações de *trazer-para-frente* e *enviar-para-trás* (ou troca de *layers*), tarefas que não podiam ser feitas antes pois as linhas ficavam fixas na `base`.

Como já mencionado, o container `AuxiliarButtons` possui os botões `Undo` e `Redo`, possibilitando a recuperação de estado do desenho em um determinado instante de tempo. É importante ressaltar que eles são válidos apenas para as ações referentes às linhas fixadas em uma determinada camada. Deste modo, se o usuário está editando máscaras da camada 2, por exemplo, ao executar o `Undo` ou `Redo`, apenas as linhas fixadas nelas serão afetadas, ao passo que as de outras camadas permanecerão inalteradas, mesmo que desenhadas em ordens prioritárias em relação ao histórico geral de edições.

A última funcionalidade a ser discutida é a escolha da resolução do *Canvas*, já presente na versão anterior, mas realocada e aprimorada na atual. No sistema antecessor, a mudança da resolução podia ser feita em qualquer momento através da seção de `Settings`. A ação, no entanto, naturalmente deformava o desenho do *Canvas*. Assim, para evitar a desconfiguração da obra, `Settings` tornou-se a primeira tela a ser exibida e pode ser editada apenas uma única vez, preservando as dimensões do quadro em que Jê Amértico estiver trabalhando. Como novidades, a nova versão ainda conta com opções pré-definidas de resoluções tanto para saídas digitais quanto físicas (dimensões de papel, no caso).

### Implementação do código

Desde sua primeira versão, houve um salto enorme no nível de complexidade do código. Todas as funcionalidades e componentes que surgiam na aplicação eram refletidos em novas estruturas no código. Assim, conforme a ferramenta foi evoluindo, a arquitetura do sistema acompanhou as mudanças e adapatou-se para poder comportar a todas elas.

A primeira grande estrutura sobre a qual a ferramenta se organizou foi a *store*, introduzida na versão da seção 4.1.2. Tratava-se de um arquivo simples que, como já explicado, continha o estado e as ações necessárias para alterá-lo. Naquele momento, tal estado consistia num objeto com os atributos: altura e comprimento do *Canvas*, um vetor de objetos que armazenavam informações das linhas a serem desenhadas e o índice da última entrada

deste *array*.

Com a evolução da ferramenta e a adição de novos componentes, maiores e mais complexos, a *store* acabou sendo dividida em estruturas denominadas módulos, cada qual responsável por um domínio funcional do programa. Com esse redimensionamento, a arquitetura do código tornou-se mais modularizada, melhorando sua clareza, organização e funcionalidade.

Sob o paradigma de programação orientada a objetos, foram criados dois arquivos módulos ou classes: `display` e `layers`. O primeiro diz respeito à configuração da tela, a saber, do tamanho do *Canvas* e do controle das seções exibidas no `MenuSection`. Assim, o módulo armazena um estado constituído pela altura e comprimento do quadro, além de um atributo que identifica a seção corrente em `MenuSection`. Já o segundo inclui os recursos referentes às camadas, implicando todos os elementos que a compõem, ou seja, `Lines` e `Masks`. Seu estado consiste em um objeto com: um vetor de objetos que reúnem todos os elementos de uma camada (nome, máscaras, linhas, etc.), um objeto para a paleta de cores, um atributo de cor para o preenchimento do fundo do *Canvas*, um índice referente a última posição do *array* de camadas, um contador de camadas, um atributo para identificar o que as linhas estão preenchendo no momento em que estão sendo editadas: máscaras ou o fundo inteiro da camada, e se os elementos dessa camada foram configurados para não serem visíveis. Deste modo, combinando os estados dos dois módulos, estabelece-se o estado geral do sistema.

Além dos módulos, a `store` contém também três classes de objetos: `color`, `lines` e `masks`. O primeiro refere-se ao componente de paleta de cores, que consiste numa lista de cores e ações de adição e atualização. O segundo consiste numa lista de objetos com informações das linhas, mais os seus métodos de manipulação. O último é composto também por uma lista de objetos com atributos e métodos referentes às máscaras. A Figura 4.13 ilustra a composição da store.

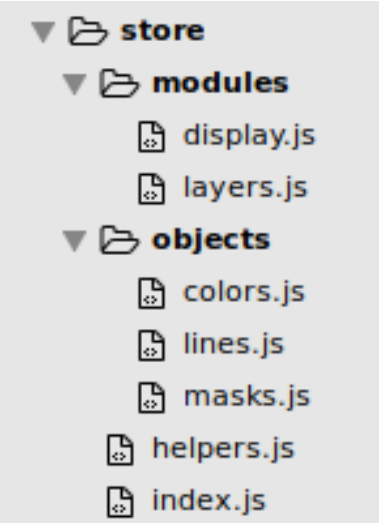



**Figura 4.13:** *Estrutura da store.*

O sistema, portanto, é estruturado a partir da reunião dos módulos e objetos citados

acima num único arquivo, `index.js`, que dispõe o estado geral do programa a todos os componentes da aplicação.

A versão final do eJê pode ser acessada através da url http://eje-arte.herokuapp.com/. A análise da aplicação e das obras provenientes dela é apresentada no capítulo seguinte.

# Capítulo 5

# Resultados

Com o término da versão oficial do *eJê*, a ferramenta precisava ser avaliada sob dois aspectos: cumprimento das expectativas em relação à proximidade das obras geradas pela ferramenta com o estilo artístico de Jê Américo, e experiência de usuário do sistema, isto é, se a aplicação proporcionava boa usabilidade e agradabilidade para a criação de novos quadros. Dessa forma, este capítulo dedica a primeira metade de seu espaço à opinião do artista sobre os resultados obtidos com o programa, ao passo que a segunda metade apresenta uma análise mais objetiva através de um modelo formal de avaliação.

## 5.1 Impressões Gerais de Jê Américo

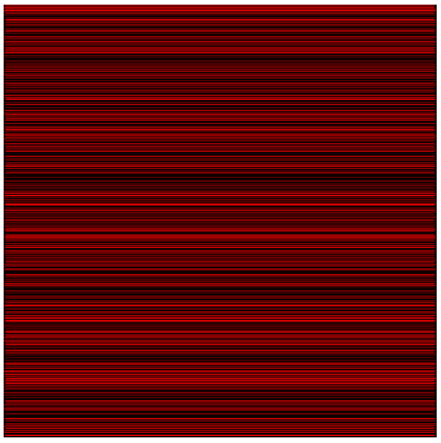
A participação de Jê Américo não se restringiu apenas ao campo de testes de usuário. Ao longo da construção do eJê, houve uma grande interação com o artista a fim de compreender seu processo criativo, condição imprescindível para determinar quais recursos a ferramenta provisionaria. Nesse sentido, a necessidade de entender as informações essenciais para viabilização da empreitada o fez refletir mais profundamente sobre os elementos utilizados na composição de seus trabalhos. Além das linhas, das formas e das cores, havia detalhes compositivos que personalizavam suas obras, como por exemplo, o desalinhamento entre o término de cada linha, recurso reproduzido pelo `Windup` no eJê.

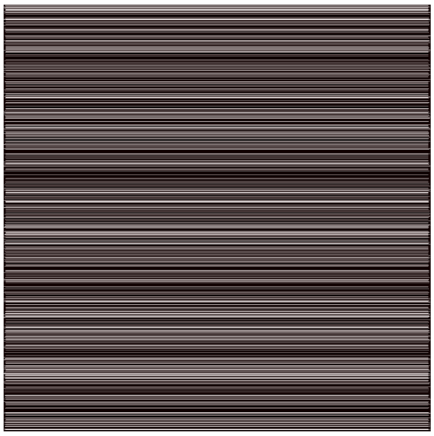
Segundo Jê Américo, a implementação de tais detalhes, na aplicação, foi de grande relevância pois permitiu recriar, com o máximo grau de fidelidade, suas produções originais e ao mesmo tempo “humanizar"os trabalhos executados digitalmente. Ademais, seus desenhos feitos à mão impõem necessariamente um limite de tempo para serem executados, pois além do trabalho manual, há também a escolha de um tema a partir do qual suas séries são originadas. Com a ferramenta, foi possível expandir essa produção de forma muito mais expressiva, uma vez que fazer uma composição é extremamente rápido através do eJê.

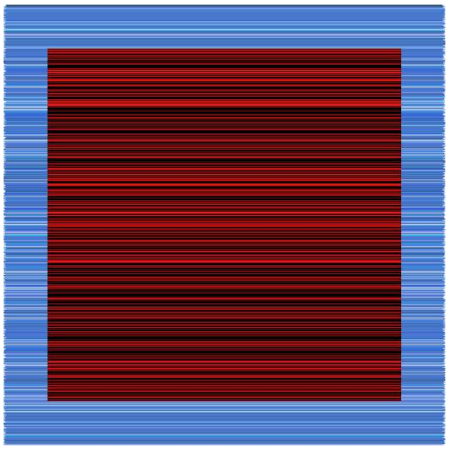
Com a ferramenta, o artista intenciona explorar uma série de possibilidades de aplicação que antes seriam impossíveis de serem executadas sem o auxílio do eJê. Ele destaca principalmente:

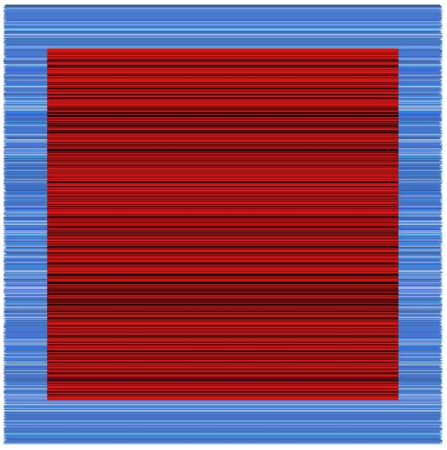
- Animações, através da produção de múltiplos frames
- Saída para produção de peças em 3D
- Mapping
- Impressões em diferentes suportes
- Alimentar o experimento de 2018 de redes neurais

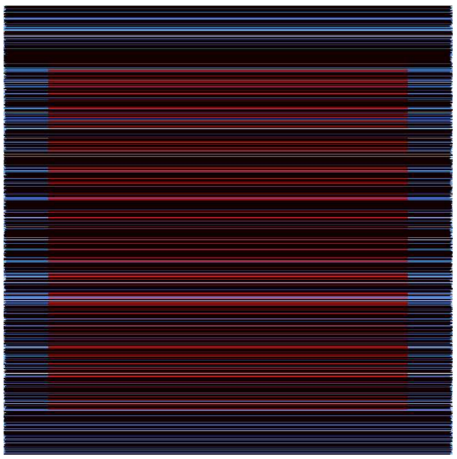
Alguns exemplos de obras geradas pelo eJê podem ser observados a seguir:

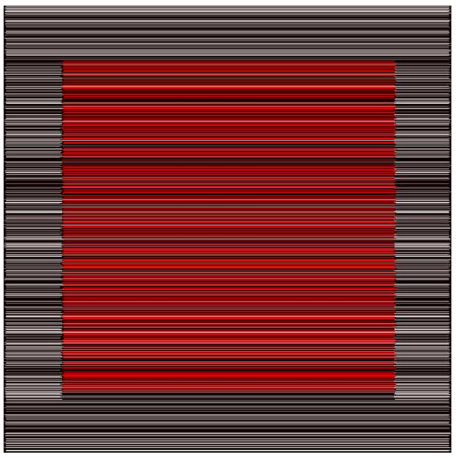


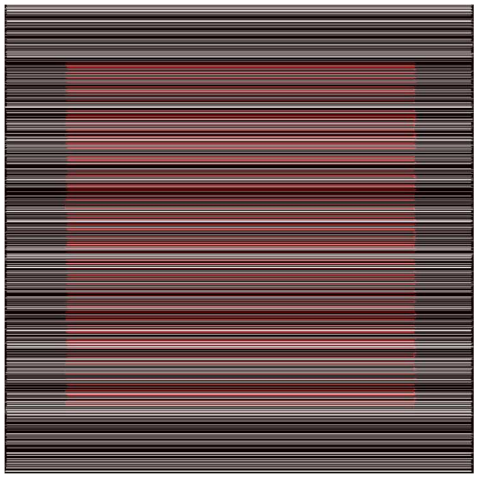


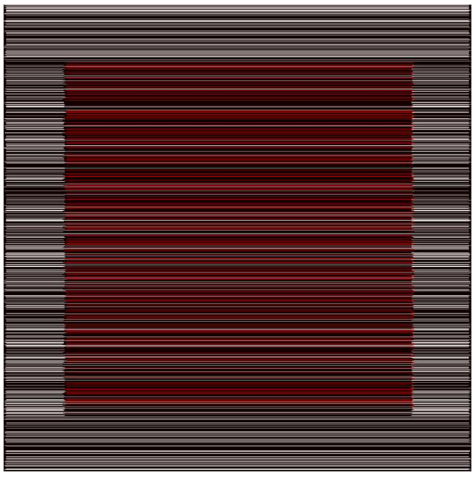


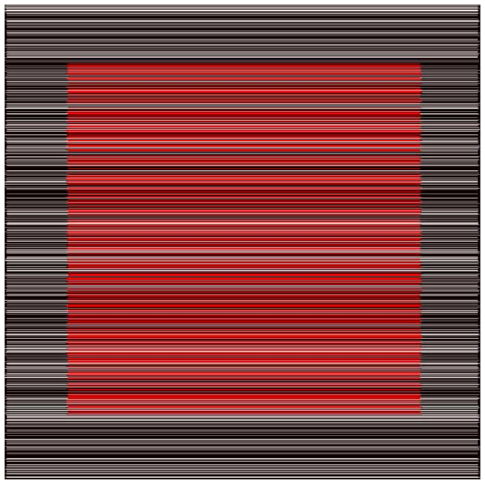


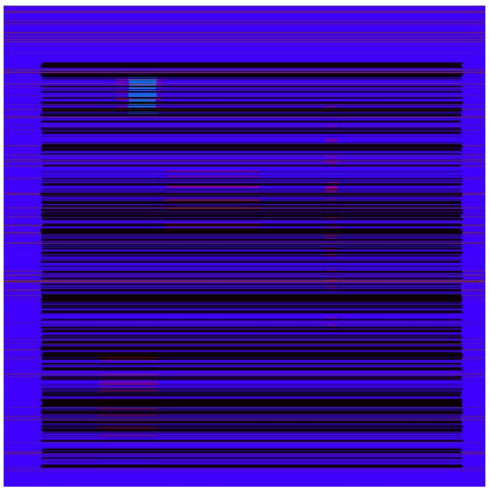


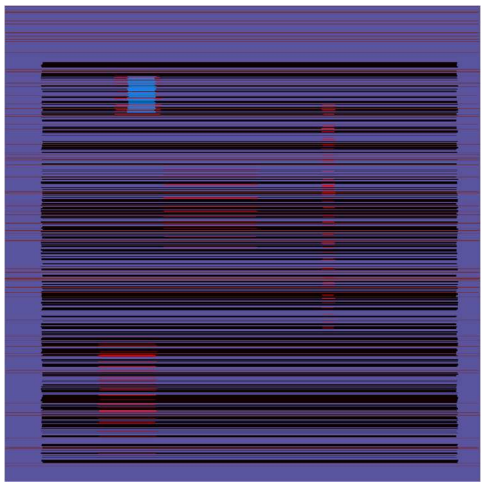


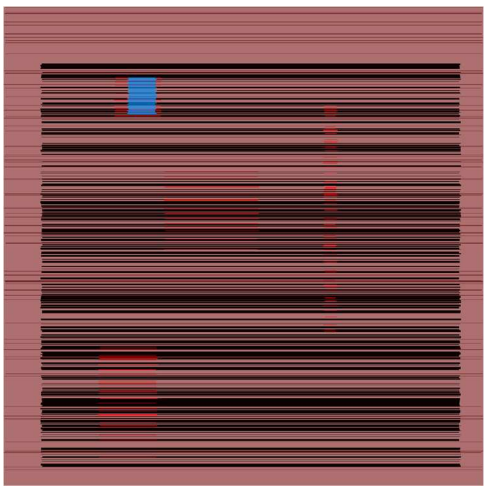


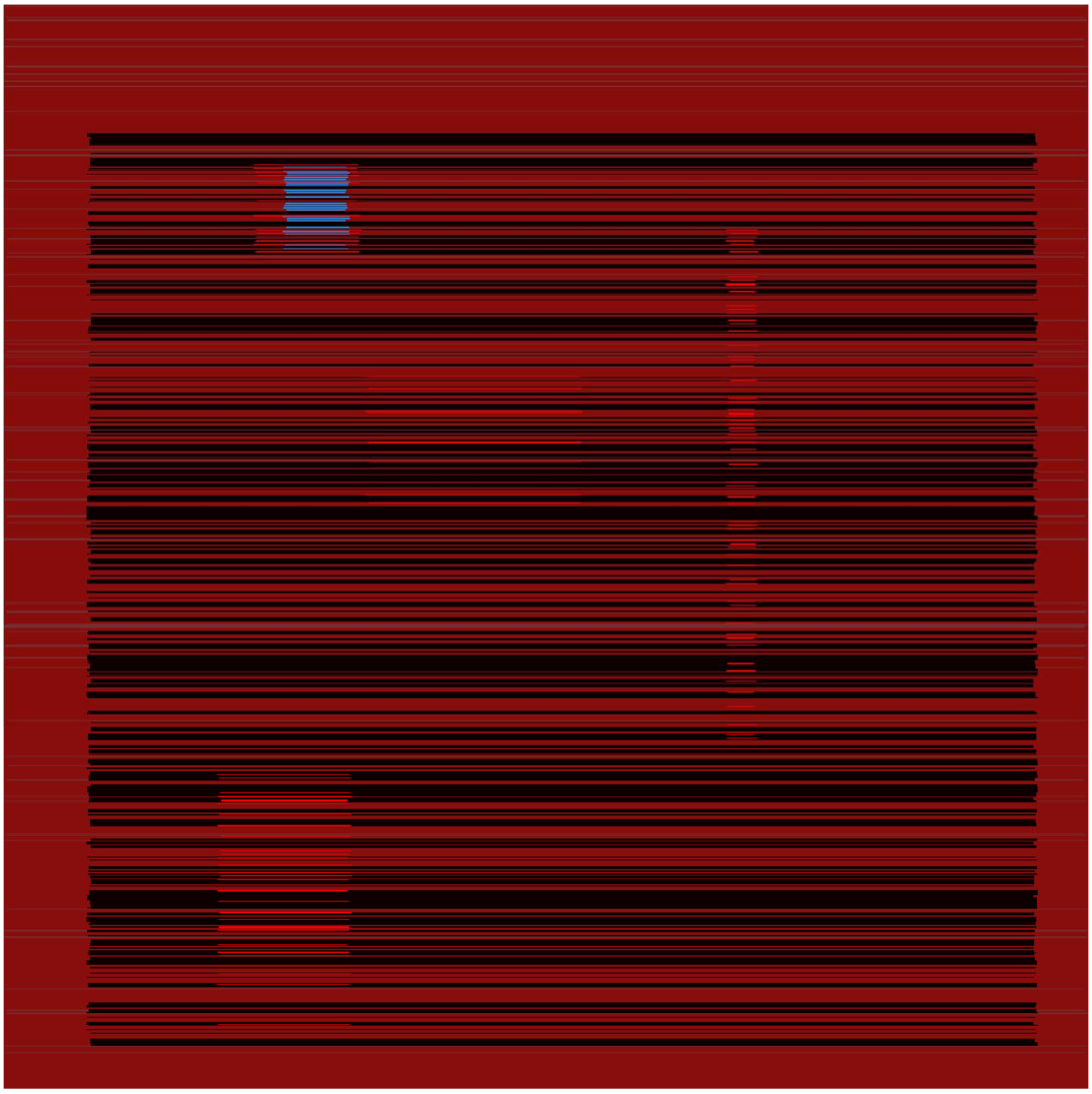












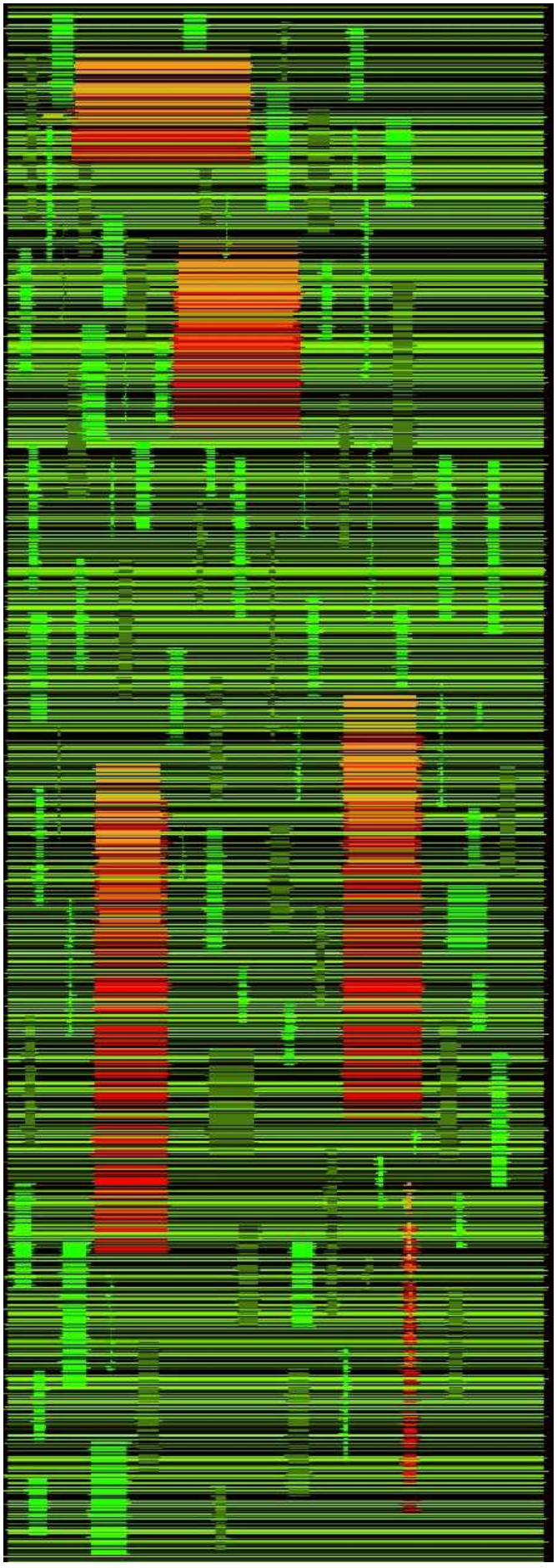


**Figura 5.1:** *Obra criada manualmente pelo artista Jê Américo apresentando um novo estilo.*

Para o artista Jê Américo, os resultados iniciais apontaram diferentes possibilidades de ricas produções e ampliaram de forma significativa os seus horizontes artísticos, mudando inclusive a forma como ele se relaciona com sua produção sobre o papel utilizando régua e canetas. A Figura 5.1 mostra uma nova obra de sua autoria com novas características estilísticas.

Por fim, o artista acredita que a interação com a ferramenta ainda apresenta questões de funcionalidade que podem ser aperfeiçoadas em futuras versões do programa, destacando:

- Inserção de imagens externas sobre os trabalhos
- Comandos de precisão (*grids*)
- Permitir o cruzamento de linhas
- Diferentes formas geométricas
- Desenho no sentido vertical

## 5.2 Avaliação

A avaliação da versão final da aplicação foi desenvolvida a partir do modelo *TAM* (*Technology Acceptance Method*, ou Modelo de Aceitação de Tecnologia) e da escala *SUS* (*System Usability Scale*, ou Escala de Usabilidade do Sistema).

*TAM* é um modelo proposto por Davis (1989) que mensura o nível de aceitação e uso efetivo de uma tecnologia. Ele consiste na análise de vários fatores referentes ao usuário e à usabilidade da tecnologia, sendo os dois principais a *PU* (*Perceived Usefulness*, ou Utilidade Percebida) e a *PEOU* (*Perceived Ease of Use*, ou Facilidade Percebida de Uso). A primeira refere-se à tendência das pessoas usarem ou não uma aplicação de acordo com o quanto elas acreditam que isso ajudará a melhorar a performance delas, enquanto a segunda, à facilidade e ao esforço do usuário para aprender a usar a aplicação.

O *SUS* (*System Usability Scale*, ou Escala de Usabilidade do Sistema) é uma escala simples criada por Brooke (1996) que permite uma perspectiva global, a partir de uma avaliação subjetiva, da usabilidade de uma aplicação. Ele é uma escala Likert focada em usabilidade constituída por 5 afirmações positivas e 5 negativas sobre o sistema, às quais o usuário deve indicar seu grau de concordância. A partir dessas respostas, é calculada uma nota para o sistema que varia entre 0 e 100.

Baseando-se nesses conceitos, foram desenvolvidas 10 afirmações focadas em *PU* e 10 focadas em *PEOU*, as quais se encontram (já preenchidas por Jê Américo) no Apêndice A, podendo ser respondidas por uma das seguintes alternativas:

1. Discordo completamente
2. Discordo parcialmente
3. Não concordo, nem discordo
4. Concordo parcialmente
5. Concordo completamente

Para cada fator da aplicação analisado, é calculada uma pontuação com valor de 0 a 100.

Para as afirmações 1, 3, 5, 7 e 9, o valor corresponde à posição da resposta na escala menos 1. Já para as afirmações 2, 4, 6, 8, e 10, o valor é 5 menos a posição da resposta na escala. Por fim, a soma de todos os valores é multiplicada por 2,5, resultando no valor total da pontuação. Assim, se o usuário responder “Não concordo, nem discordo” para todas as afirmações, a pontuação final da aplicação será 50, sendo desejável assim uma pontuação maior que esse valor.

Além disso, comparando-se as afirmações desenvolvidas com as perguntas propostas a serem usadas com o *SUS* (Brooke, 1996), pode-se ver que há muitas semelhanças e foco na facilidade de uso da aplicação. Desse modo, para avaliar a *PEOU*, também se comparou a sua pontuação final com as de outros sistemas que aplicaram o *SUS*. De acordo com Bangor, Kortum e Miller (2009), a média de pontos para um sistema considerado “Bom” é 71,4, sendo essa classificação a 3.a melhor em um sistema de 7 posições.

O sistema eJê obteve respectivamente as pontuações 92,5 e 87,5 referentes a *PU* e *PEOU*, estando, portanto, acima da média e podendo inclusive ser classificado como “Excelente"(cuja

média de pontos é 85,5). Dessa forma, pode-se concluir que a aplicação possui usabilidade muito boa e cumpre com grande êxito o propósito atribuído a sua funcionalidade, sendo uma ferramenta de fácil uso e extrema utilidade para Jê Américo.

# Capítulo 6

# Conclusões

A construção de um sistema compreende desafios que vão muito além de problemas de programação ou implementação de código. É um trabalho que demanda organização, planejamento e estudo, pois as decisões de projeto são fatores que podem impactar o programa em qualquer etapa do processo de desenvolvimento, compromentendo seu avanço e desempenho. Nesse sentido, a aplicação de metodologias e práticas voltadas à sistematização do trabalho foram essenciais para a evolução da ferramenta, uma vez que proporcionaram um controle operacional sobre as tarefas que deviam ser cumpridas. Logo, destaca-se aqui o papel fundamental da implantação de métodos do *scrum*, em especial, as *Sprints*.

A definição destas *Sprints* junto à orientação dos supervisores do projeto tornavam as atividades mais direcionadas e alinhadas com as prioridades do momento, além de ajudar na administração do tempo. Portanto, a consistência da organização e discussão de tarefas é imprescindível para o bom andamento da produção. Há, porém, um outro aspecto indispensável ao desenvolvimento de um sistema e que nem sempre acompanha a ideia de um trabalho metódico: a flexibilidade.

Durante a construção da ferramenta, inúmeros imprevistos surgiram, desafios inesperados foram encontrados e necessidades novas e mais urgentes nasciam. Assim, a flexibilização foi uma característica fundamental para contornar problemas de natureza imprevisível e adaptar o trabalho às prioridades do projeto. Consequentemente, alguns planejamentos e reuniões fugiam do esperado, mas eram investidos o máximo de esforço e resiliência para realinhar as atividades.

Um aspecto muito positivo ao longo do desenvolvimento da aplicação e que se mostrou sempre favorável à resolução dos problemas foi a escolha das tecnologias, com grandes méritos ao *Vue.js* e ao *HTML Canvas*. Ambos forneceram todos os subsídios necessários para a implementação das funcionalidades da ferramenta, sendo que as maiores dificuldades encontradas eram decorrentes mais da lógica de programação do que de limitações da linguagem.

Apesar de equipados com bons recursos tecnológicos e seguindo, na medida do possível, uma agenda de tarefas com prazos de entrega, a ferramenta não foi finalizada com todas as funções definidas desde o seu planejamento inicial. O tempo limitado para o cumprimento

das atividades acarretou o preterimento de funcionalidades que seriam muito interessantes ao artista Jê Américo na criação de obras.

Entre as funções não entregues destacam-se a introdução de máscaras em outros formatos geométricos, a opção de linhas verticais e mais dois recursos que representam os maiores decréscimos em relação à proposta inicial da aplicação: a animação das obras e a modelagem 3D dos quadros.

Uma das ideias primordiais da ferramenta era prover um meio para atribuir movimento aos elementos de uma composição, ou seja, animar a obra, recurso de grande interesse ao artista, uma vez que as limitações do árduo trabalho manual por trás de suas produçõees não permitiam a elaboração de quadros em quantidade abundante para a definição de *frames* de um vídeo. O *eJê*, entretanto, oferece uma funcionalidade que visa auxiliar a produção de inúmeras imagens com pequenas alterações nas distribuições das linhas, imagens estas que poderiam ser dispostas em sequência (através de um editor externo de vídeos), criando a ilusão de movimento.

Já a modelagem 3D representava uma tentativa de conceber obras em alto-relevo para tornar os quadros de Jê Américo acessíveis a deficientes visuais. Sendo talvez a maior perda, a ferramenta não possui nenhum tipo de função para esta tarefa e acabou não trazendo consigo nenhum tipo de suporte para acessibilidade.

Os recursos citados acima, apesar de inexistentes na versão oficial do *eJê*, representam possíveis projetos futuros da ferramenta, sendo o último, inclusive, uma enorme oportunidade de estudo e desenvolvimento sobre um tema com extremo potencial: arte e acessibilidade.

Apesar da ausência de algumas funcionalidades, a ferramenta obteve grande êxito no cumprimento de seu objetivo primordial: reproduzir os elementos das obras de Jê Américo a fim de auxiliá-lo na produção de novas, de maneira rápida, fácil e intuitiva. Com ótimas pontuações calculadas a partir dos fatores *PU* e *PEOU* (92.5 e 87.5, respectivamente), pode-se afirmar que o sistema possui usabilidade muito boa e executa com sucesso as tarefas que se propoõe a fazer, sendo de grande utilidade e relevância ao artista.

De modo geral, as obras geradas pela ferramenta não só conseguiram recriar com fidelidade os traços característicos da estética de Jê Américo como também expandiram seus horizontes como artista. O eJê abre um enorme espaço para a experimentação com os recursos que dispõe e proporciona novas possibilidades de experiências visuais e estilos, superando as limitações criativas decorrentes do simples uso de papel, caneta e régua. Dessa forma, demonstra-se como a tecnologia pode ser uma grande aliada ao processo de criação, conferindo novas formas de produção e manifestação artísticas.

# Apêndice A

# Métodos de Avaliação

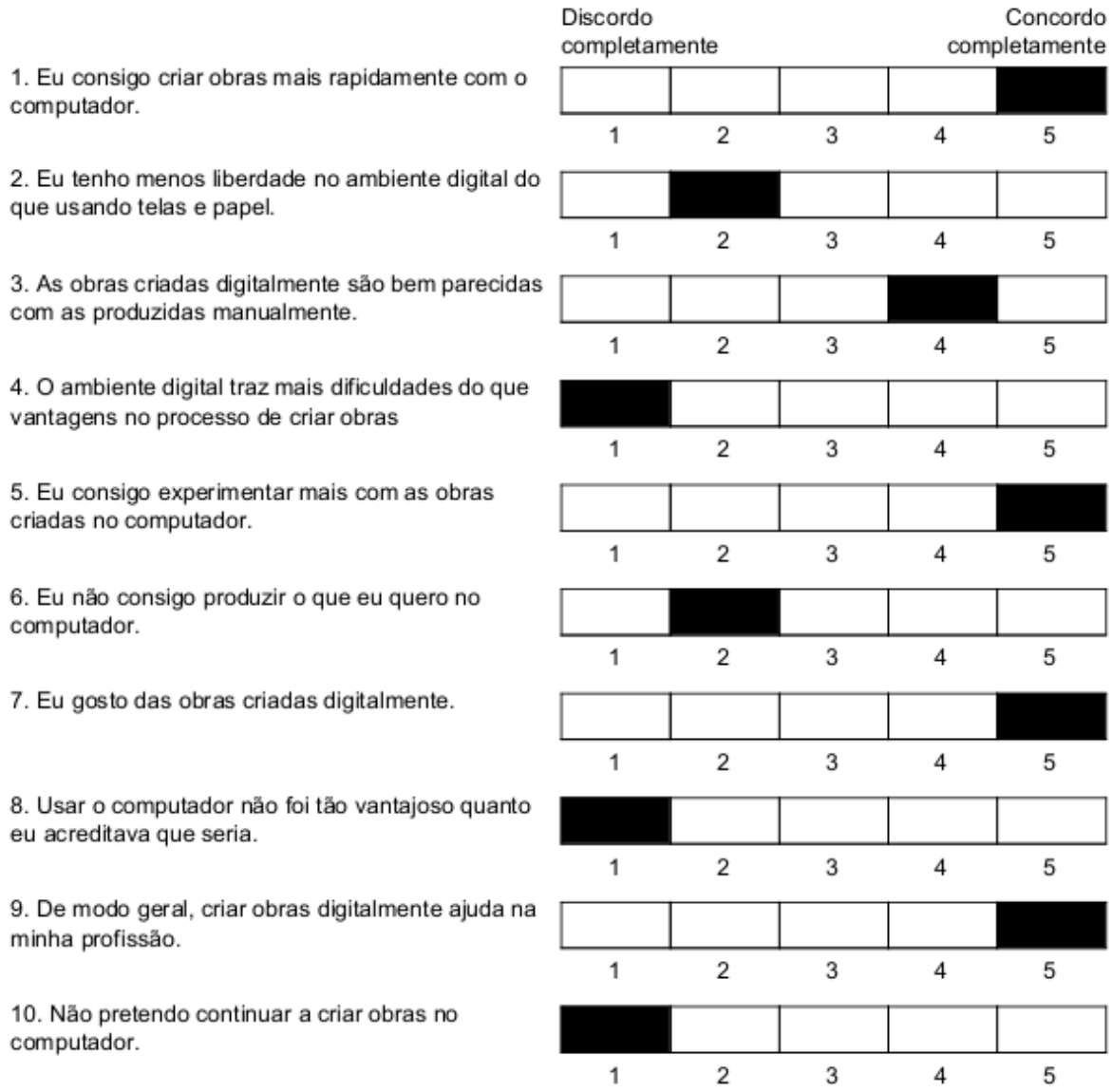



**Figura A.1:** *Afirmações para a avaliação de PU*

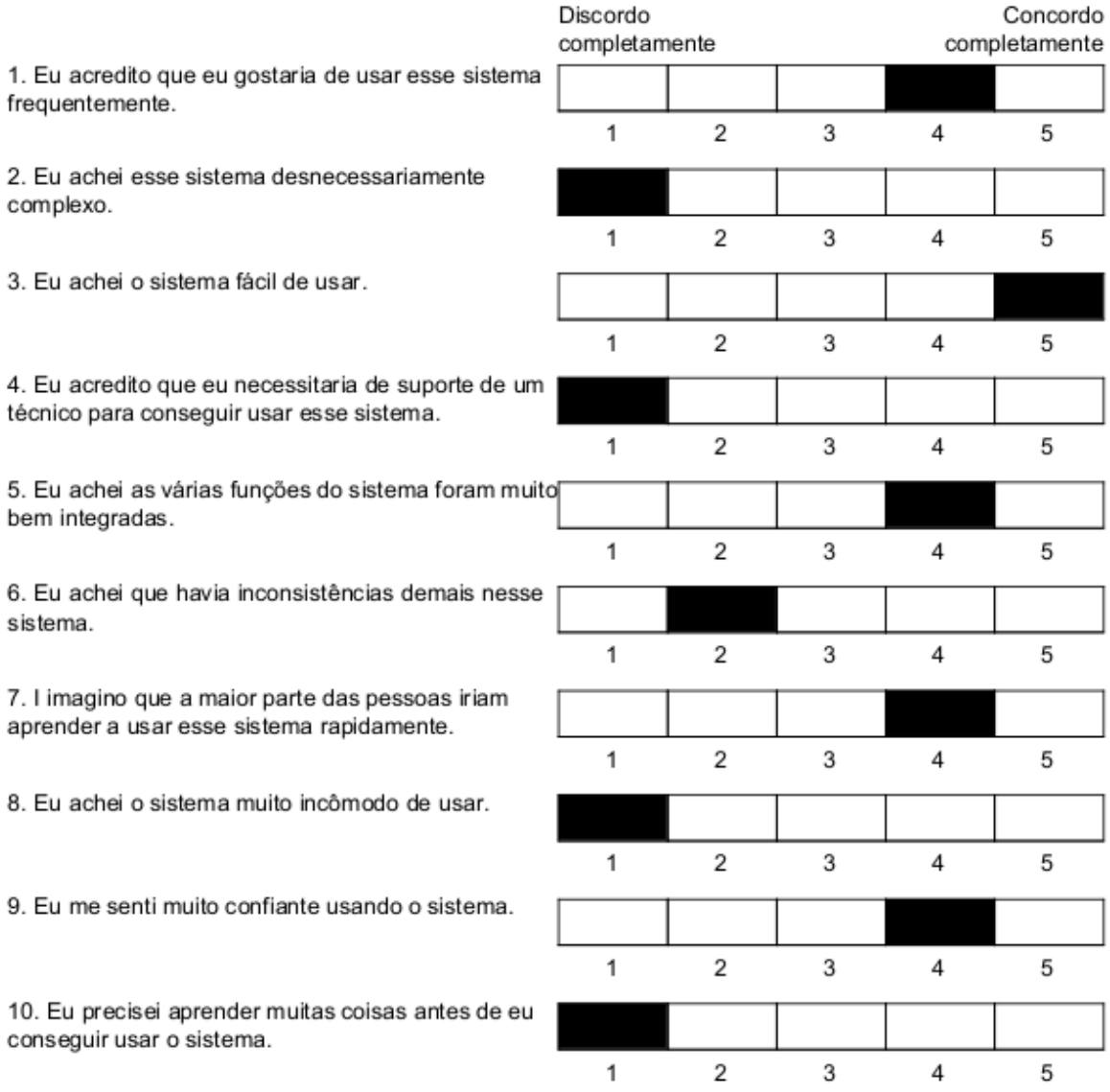



**Figura A.2:** *Afirmações para a avaliação de PEOU*

# Referências Bibliográficas


**Bangor** *et al.***(2009)** Aaron Bangor, Philip Kortum e James Miller. Determining what individual sus scores mean: Adding an adjective rating scale. *Journal of Usability Studies*, 4(3):114–123. URL http://uxpajournal.org/wp-content/uploads/sites/8/pdf/JUS_Bangor_May2009.pdf. Citado na pág. 36

**Brooke(1996)** John Brooke. Sus - a quick and dirty usability scale. https://hell.meiert.org/core/pdf/sus.pdf, 1996. Último acesso em 25/11/2019. Citado na pág. 36

**Davis(1989)** Fred D. Davis. Perceived usefulness, perceived ease of use, and user acceptance of information. *MIS Quarterly*, 13(3):319–340. URL https://www.jstor.org/stable/pdf/249008.pdf. Citado na pág. 36

**Espinha(2019)** Roberto G. Espinha. Kanban: Aprendendo a gerenciar fluxos de trabalho. https://artia.com/biblioteca/pdf-kanban/, 2019. Último acesso em 30/10/2019. Citado na pág. 3

**Fulton e Fulton(2013)** Steve Fulton e Jeff Fulton. *HTML5 Canvas*. O'Reilly, Sebastopol, USA, 2 edição. Citado na pág. 10

**Galdino(2017)** Fabricio Galdino. Vue.js tutorial. https://www.devmedia.com.br/vue-js-tutorial/38042, 2017. Último acesso em 31/11/2019. Citado na pág. 10

**Mendonça(2018)** Isabella C. F. Mendonça. redes neurais convolucionais gerando arte: aplicação em obras de jê américo. Relatório técnico, Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo, Brasil. URL https://www.linux.ime.usp.br/~adnan/mac0499/misc/monografia.pdf. Citado na pág. 2

**Rizolli(2013)** Marcos Rizolli. Percursos estruturantes da imagem digital fixa. vera molnar e laurence gartel: dois artistas, um estudo. Em *ENCONTRO INTERNACIONAL DE ARTE E TECNOLOGIA*, 12, Distrito Federal. URL https://art.medialab.ufg.br/up/779/o/art13_MarcosRizolli.pdf. Citado na pág. 1

**Schwaber e Sutherland(2017)** Ken Schwaber e Jeff Sutherland. *The Scrum Guide*, 2017. URL https://www.scrumguides.org/docs/scrumguide/v2017/2017-Scrum-Guide-US.pdf. Citado na pág. 3

**Schwartz(1995)** Lillian Schwartz. The art historian's computer. *Scientific American*, 272 (4):106–111. URL https://www.jstor.org/stable/24980579?seq=1. Citado na pág. 1

**Vianna** *et al.***(2012)** Maurício Vianna, Ysmar Vianna, Isabel K. Adler, Brenda Lucena e Beatriz Russo. *Design Thinking: Inovação em negócios*. MJV Press, Rio de Janeiro, Brasil. Citado na pág. 5