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

# Prova de conceito de aplicação de desenho vetorial controlada pelo olhar

Veronica Maria Sarti Stocco

Monografia Final

mac 499 — Trabalho de Formatura Supervisionado

Supervisor: Prof. Dr. Carlos Hitoshi Morimoto

São Paulo
2021

*The real interaction designer's decisions are based on what the user is trying to achieve.*
*Alan Cooper*

# Agradecimentos

*All moments, past, present and future, always have existed, always will exist. [...] It is just an illusion we have here on Earth that one moment follows another one, like beads on a string, and that once a moment is gone it is gone forever.*
— Kurt Vonnegut

Ao professor Carlos Hitoshi Morimoto, pela orientação, conselhos e inspiração ao longo dos anos desde que cursei a disciplina de Interação Homem-Computador.

Ao professor Marco Dimas Gubitoso, pela supervisão, apoio e conversas sem as quais certamente não haveria chegado até aqui.

À minha família e amigos, cujo apoio incondicional ao longo da graduação tornaram tudo isso possível.

# Resumo

Veronica Maria Sarti Stocco. **Prova de conceito de aplicação de desenho vetorial controlada pelo olhar**. Monografia (Bacharelado). Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo, 2021.

O presente trabalho teve como objetivo desenvolver uma prova de conceito de uma ferramenta de desenho vetorial controlada pelo olhar. Tal ferramenta permitiria maior acessibilidade para a geração de ilustrações. Para tanto, realizou-se um estudo de design de interação focado no usuário. Por meio da análise de ferramentas já existentes e da criação de diversos protótipos de baixa fidelidade, foi possível criar um protótipo final. Utilizando como base uma ferramenta de desenho vetorial chamada Dotgrid, foram realizadas alterações em sua interface para torná-la compatível com o uso de eye trackers.

**Palavras-chave:** Interação Homem-Computador. GazeBar. Desenho vetorial. Troca de contexto. Design de interação. Interação pelo olhar. Dotgrid.

# Abstract

Veronica Maria Sarti Stocco. **Proof of concept of a vector graphics application controled by the gaze**. Capstone Project Report (Bachelor). Institute of Mathematics and Statistics, University of São Paulo, São Paulo, 2021.

The goal of the current work was to develop a proof of concept of a vector graphic drawing tool controlled by gaze interaction. Such a tool would allow greater accessibility when it comes to creating illustrations. With that goal, an interaction design study was conducted, focusing on the end user. Through the analysis of preexisting tools and the creation of several low-fidelity prototypes, it was possible to create a final prototype. A vector graphic tool named Dotgrid was used as a basis. Modifications were made to its interface to make it compatible with the use of eye trackers.

**Keywords:** Human-Computer Interaction. GazeBar. Vector graphics. Context switching. Interaction design. Gaze interaction. Dotgrid.

# Sumário

# Capítulo 1

# Introdução

O método principal de interação de um usuário com um computador se dá por meio do uso de mouse e teclado. Consequentemente, a maioria dos programas existentes exigem considerável coordenação motora para serem operados. Como tal, são inacessíveis para diversas pessoas. Em tais casos, métodos de interação pelo olhar são preferíveis.

Diversas ferramentas onde seleções são controladas pelo olhar já foram desenvolvidas. Entretanto, existem poucas quer permitem a criação de ilustrações. O seguinte trabalho desenvolvido ao longo de 2021, e visou estudar e propor uma solução para a implementação de uma ferramenta de desenho vetorial controlada pelo olhar.

## 1.1 Conceitos utilizados no trabalho

Esta seção contem a definição de conceitos que serão referenciados diversas vezes ao longo deste trabalho, e que também são utilizados para definir a prova de conceito a ser desenvolvida.

### 1.1.1 Controle pelo olhar

A interação com o olhar é baseada em medições do movimento dos olhos e dos pontos nos quais eles estão focados. Para obter tais medidas com precisão, é necessário o uso de equipamentos de *eye tracking*. Ferramentas controladas pelo olhar podem ou não requerir um processo de calibramento do usuário antes do início da sessão de uso. Restrições de tempo e a pandemia impossibilitaram a familiarização com tais equipamentos. O presente trabalha tem como foco os conceitos teóricos de técnicas de controle pelo olhar.

Em tal forma de interação, seleções podem ser realizadas de duas formas principais: via *gaze dwelling* ou mudança de contexto.

#### *Gaze dwelling*

Uma seleção por gaze dwelling é realizada quando um usuário mantém o olhar em determinada área por um período pré-determinado de tempo, chamado de *dwell time*. Após

a passagem do *dwell time*, o sistema retorna um feedback visual, confirmando a seleção. Apesar de ser uma forma "intuitiva"de realizar tal interação, ela pode introduzir diferentes fontes de frustação para usuários. A espera de um dwell time longo pode ser incômoda para usuários experientes. Para novatos, um *dwell time* curto pode resultar em seleções erradas. A ocorrência de tal tipo de seleção não desejada recebe o nome de Problema de Toque de Midas.(Carlos H. Morimoto *et al.*, 2018)

**Mudança de contexto**

Este método faz proveito do problema do toque de Midas. Ao focar o olhar em uma opção, há um feedback visual instantâneo. Entretanto, a seleção só é confirmada quando o usuário muda o olhar para outra região do programa. Por exemplo, deixando a área de um menu e atingindo a área de trabalho. Isso permite que usuários possam explorar as opções disponível pelo tempo que desejarem, sem correr o risco de realizar uma seleção acidental. Ademais, usuários experientes podem realizar seleções de forma rápida, sem a limitação de aguardar por um *dwell time* imposto pelo sistema.

O GazeBar é um método de interação com o olhar desenvolvido por Elmadjian e Morimoto em 2021. Trata-se da implementação de um menu primário, e possíveis menus secundários e terciários, que permitem a realização de seleções por meio de mudança de contexto (Elmadjian e Carlos H Morimoto, 2021).

### 1.1.2 Formas de desenho

Em ferramentas de ilustração, duas das mais comuns formas de desenho são pelo uso de pincéis e de ferramentas que permitem a criação de vetores. Pincéis tradicionalmente permitem a criação de imagens rasterizadas por meio de desenho livre, controlado pelo mouse ou por um tablet. Apesar de ser possível utilizar o olhar como *input* para tal tipo de desenho, observou-se que o controle sobre tais ferramentas é difícil, e não é indicado. (Hornof *et al.*, s.d.).

Ferramentas de ilustração como o Adobe Photoshop e Illustrator entendem vetores como um objeto definido por pontos e as retas ou curvas que os unem. Isso permite fracionar o processo de desenho em etapas distintas: a seleção de um ponto, do ponto seguinte, e de como devem ser unidos. Tal método apresenta maior compatibilidade com as limitações relativas à interação com o olhar e, por esse motivo, é o escolhido para o desenvolvimento do projeto.

### 1.1.3 Imagens vetoriais

Uma imagem definida por meio de um vetor poderia, ainda assim, ser rasterizada. Optou-se por trabalhar com imagens vetoriais, principalmente por estas serem escaláveis (é possível aumentar seu tamanho sem perda de resolução).

### 1.1.4 *Flow*

Se entende como *flow* o estado de foco no qual uma determinada tarefa pode ser realizada com fluidez. (Csikszentmihalyi, 2014) O estado de *flow* é interrompido quando

existe alguma fricção decorrente da interação com o sistema. O estudo de design de interação realizado foca em identificar e minimizar as possíveis pontes de fricção de interação.

## 1.2 Objetivos

O trabalho em questão visou implementar uma prova de conceito de uma aplicação de desenho vetorial controlada pelo olhar, fazendo uso do GazeBar para realizar as seleções.

Foram utilizadas técnicas de Interação Homem-Computador para garantir que a proposta final permita que o usuário mantenha o *flow* durante sua operação. De tal forma, o objetivo não é apenas idealizar uma ferramenta que possa ser controlada pelo olhar, mas sim uma que maximize o conforto e facilidade do usuário ao operá-la.

# Capítulo 2

# Princípios de Interação Homem-Computador no contexto do trabalho

Alan Cooper define *cognitive friction* como "a resistência encontrada por um intelecto humano quando ele engaja com um sistema complexo de regras que mudam conforme o problema permuta". (Alan Cooper, s.d.) Tal fricção diz respeito não somente à dificuldade de compreender um sistema em um primeiro contato, mas de ser capaz de operar o mesmo após um dado período de tempo. Esse problema é especialmente frequente em softwares, onde uma mesma ação ou comando podem levar a respostas distintas.

A aplicação de princípios de IHC desde o início do processo visa incorporar o foco no design em todas as etapas do processo. Quando o design só é considerado no desenvolvimento da interface final, há um limite do que ele pode alterar. Elementos cosméticos são essenciais para garantir uma boa experiência do usuário. Entretanto, não é tão fácil corrigir fricções de interação intrínsecas ao sistema. (Alan Cooper, s.d.) Isso acontece porque a interação com o sistema não se restringe à aparência de um dado botão pressionado para realizar uma ação, mas sim à totalidade de ações necessárias para realizar uma tarefa, o tempo demorado para executá-las e os feedbacks oferecidos pelo sistema durante a interação. Visou-se desenvolver uma proposta de sistema que atenda às necessidades do usuário.

## 2.1 Estudo de ferramentas já existentes

Dois programas de desenho com o olhar foram analisados, e estão detalhados nas subseções a seguir.

### 2.1.1 EyeSketch

O programa permite a criação e edição de desenhos utilizando o olhar. Ele oferece três ferramentas de desenho: retângulo, oval e reta. Não é possível criar curvas, ou um único objeto mais complexo. A seleção de ferramentas é realizada por meio de *gaze dwelling*,

com um *dwel time* descrito como "relativamente curto". Logo, cada etapa do processo de escolher uma forma, posicioná-la no *canvas* e definir sua cor é limitada pelo tempo de espera para que o sensor identifique que uma ação foi confirmada.

É possível alterar a cor, tamanho e posição de um retângulo ou oval após sua criação. Para isso, é preciso olhar para o mesmo por um *dwel time* para selecioná-lo. Em seguida, é possível realizar gestos com o olhar para modificar o objeto. A movimentação só é possível em oito eixos predefinidos. (Heikkilä, 2013)

Ao se observar a imagem desenhada, é possível selecionar algum dos objetos criados por acidente (um exemplo do Problema do Toque de Midas). Para minimizar a chance de tais erros, o programa apresenta um modo chamado *look around*. Ativado por um botão, ele bloqueia a possibilidade de se selecionar um dos objetos que integra a imagem. Desta forma, é possível observar a ilustração criada sem o risco de realizar uma seleção. Existe uma opção para habilitar ou desabilitar a visualização de uma *grid*: uma série de retas paralelas equidistantes, no sentido vertical e horizontal, que auxiliam a posicionar os objetos no *canvas*.

### 2.1.2 EyeDraw

A ferramenta foi criada com o foco de permitir que crianças com limitações motoras desenhassem. Similarmente ao EyeSketch, ela também permite definir retângulos, ovais e retas. Ademais, também permite "estampar"uma ilustração previamente criada (como um *clip-art*) sobre a imagem. É possível colorir as figuras com cores predefinidas. A interação é completamente baseada em *dwell time*. Não é possível alterar uma figura depois de sua criação. O EyeDraw também permite habilitar ou desabilitar a visualização de uma *grid* para auxiliar o processo de desenho. (Hornof *et al.*, s.d.)

Ambas ferramentas possuem ferramentas similares, e sua análise permitiu identificar pontos de melhoria. Apesar de ser possível criar desenhos complexos com ambas, existe atrito na interação ao se tentar criar figuras compostas. Considere o caso ilustrado abaixo, em que se deseja ilustrar uma nuvem. A figura (a) representa o resultado final desejado. O contorno preto em (b) e (c) ilustra como a imagem final foi composta.

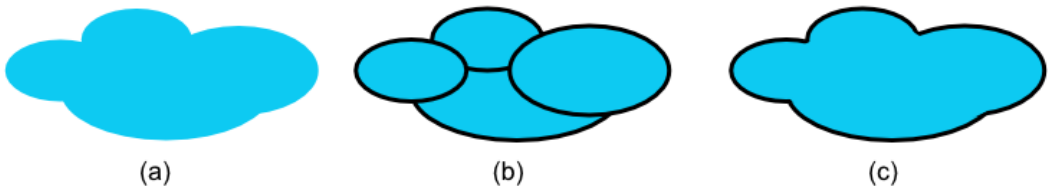


**Figura 2.1:** *O resultado visual que se deseja atingir é (a): uma representação de uma nuvem. Em (b), a nuvem foi composta por quatro formas ovais distintas. Já em (c), ela é um objeto composto por uma série de pontos e curvas que os unem.*

É possível obter a ilustração da nuvem no EyeSketch por meio da criação de quatro formas ovais (b). Entretanto, caso seja necessário movê-la, ou alterar sua cor, seria necessário repetir a sequência de ações para cada um dos quatro círculos que a compõe. Tal

interação teria muito menos atrito caso fosse possível gerar uma imagem como aquela em (c), composta por uma série de pontos e curvas. Para modificar a posição ou cor de (c), bastaria realizar a tarefa uma única vez.

A principal melhoria identificada é a diminuição de atrito ao realizar a interação por meio de mudança de contexto ao invés de *dwell time*. Isso permitiria que o desenho pudesse ser realizado de forma mais simples e fluida, de modo a facilitar que o usuário final possa entrar em estado de *flow* e não sentir a interação pelo olhar como um empecilho.

## 2.2 Prototipação

O processo de design para criação de uma solução não é linear. Diversas alternativas são consideradas antes de se atingir aquela considerada a ideal. A criação de protótipos de baixa fidelidade é uma forma se explorar possíveis alternativas. Tais protótipos podem ser criados em folha de papel. O intuito não é somente esboçar a interface gráfica da aplicação, mas também representar o processo envolvido em realizar uma ação. (Experience, 2022), (Dam e Siang, 2022)

Foi elaborada uma lista de tarefas que a ferramenta ideal possuiria:

- relacionadas a vértices: criar, selecionar, mover e deletar;
- relacionadas a polígonos: criar, selecionar, mover e deletar;
- definir e alterar a cor de preenchimento de um objeto;
- definir e alterar a cor e tamanho da borda de um objeto;
- desfazer ação anterior;
- refazer ação anterior

Entre todas,considerou-se que a tarefa base a ser realizada é a definição de um ponto. Apesar de aparentemente simples, diversas iterações do processo de prototipação tornaram claro que o processo envolvia diversos estágios.

A primeira solução considerada, representada abaixado, foi baseada no uso de um *grid*. Ela representa a etapa final da criação de um triângulo por meio da definição de seus três pontos. Para tanto, era necessário selecionar a ferramenta de criação de retas no GazeBar. Em seguida, o usuário deveria selecionar com o olhar um ponto nas intersecções de retas do *grid*. Tal solução seria um híbrido de seleção por mudança de contexto e por *dwell time*, e por esse motivo foi descartada.

O protótipo inicial evidenciou que o processo para selecionar diversos pontos somente por meio de mudança de contexto não seria um processo simples. Uma das principais limitações desse exemplo é o número limitado de pontos que podem integrar um desenho.

A solução escolhida para tal problema foi o uso de barras de rolagem, horizontal e vertical, que permitam selecionar as coordenadas exatas que o ponto deva ter na tela. Ainda assim, devido à precisão limitada de *eye trackers*, a quantidade de possíveis pontos seria limitada.

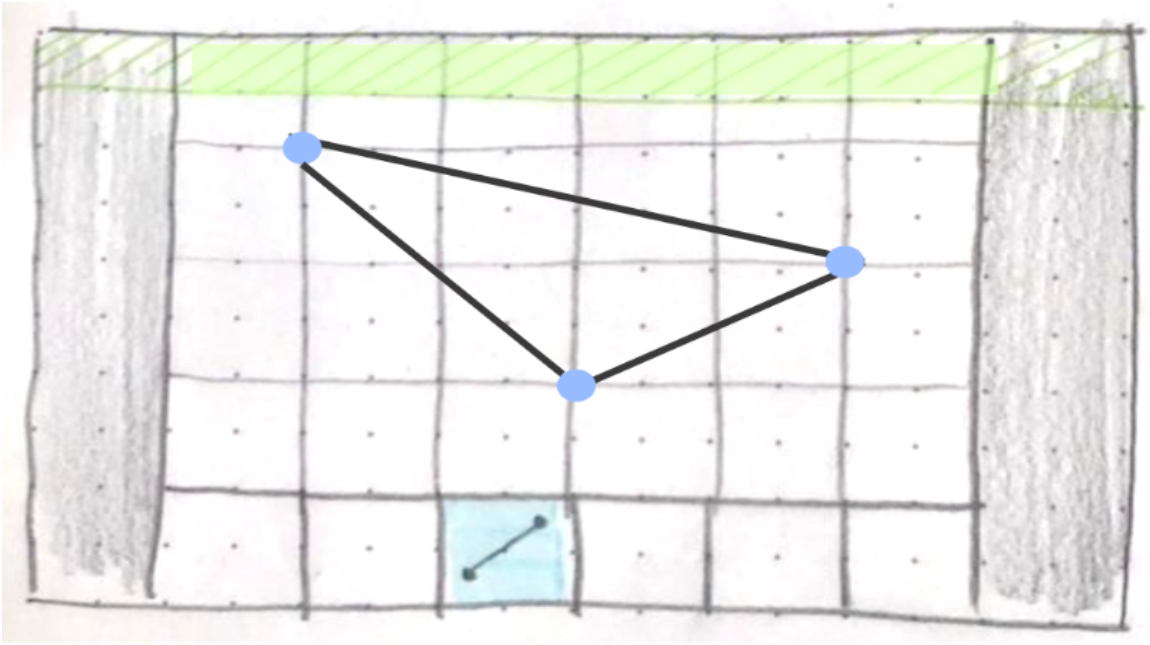

**Figura 2.2:** *Representação do primeiro protótipo criado.*

Desta forma, se identificou a necessidade de implementar uma ferramenta de zoom, que permitiria selecionar as coordenadas desejadas com maior precisão.

Diversas iterações de protótipos de papel foram criadas ao longo de alguns meses, até atingir o protótipo final, representado a seguir.

Nas ilustrações a seguir, a imagem central representa a área de desenho. A barra inferior se trata do GazeBar, que apresenta as opções pertinentes ao processo de definir um ponto. As opções apresentadas no GazeBar podem ser alternadas a depender da tarefa que se está realizando no momento. A barra da esquerda representa o zoom, e as da direita e do topo são utilizadas para mover o foco da tela. O funcionamento de tais barras de rolagem será ilustrado nas imagens (4) e (5).

O ponto central é identificado pelo perímetro de um círculo, de forma a permitir que o usuário visualize exatamente qual ponto está sendo selecionado. Não é possível "arrastar"a imagem — só é possível transladar vertical ou horizontalmente, sempre mantendo o círculo que representa o ponto a ser selecionado no centro da tela.

Em (1), vê-se o estado inicial de definição de um ponto. O zoom está em seu valor mínimo, e a imagem se encontra centralizada. A ação inicial (2) é a seleção da barra de zoom, à esquerda. Ela tem sua cor alterada como feedback visual imediato.

Com a barra selecionada, ela é movimentada para cima (3), de modo a aumentar o zoom. Entre (3) e (4), o olhar retorna para o centro da tela, de modo a finalizar a alteração de zoom. Em (4), a barra de deslocamento horizontal é selecionada. Sua cor é alterada como feedback visual, e uma linha horizontal percorre a tela. O intuito disso é oferecer uma resposta visual indicando que a ação desta barra é distinta daquela de zoom.

Movendo o olhar para a esquerda (5), a imagem do barco é transladada de acordo. Após

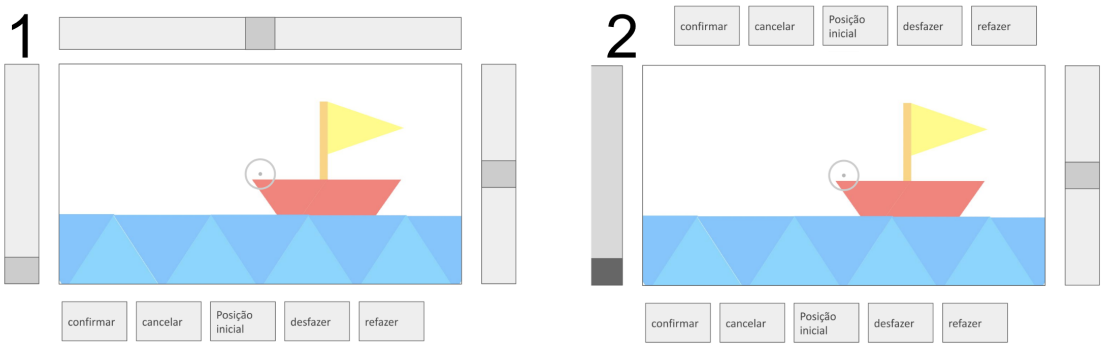


**Figura 2.3**

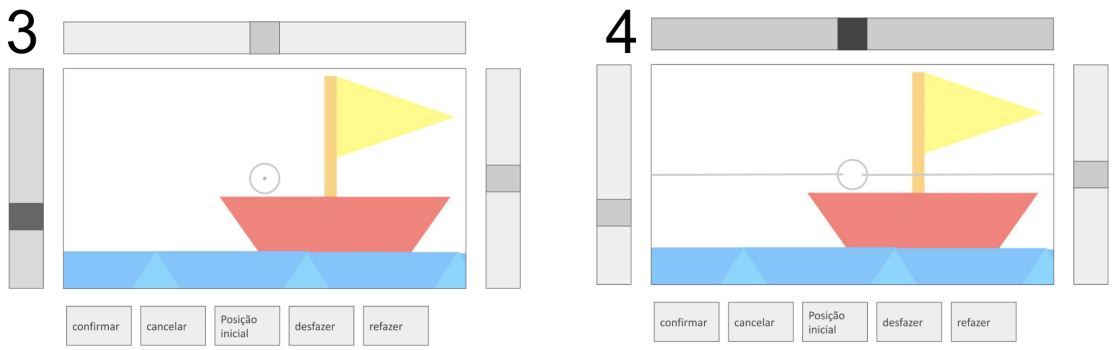


**Figura 2.4**

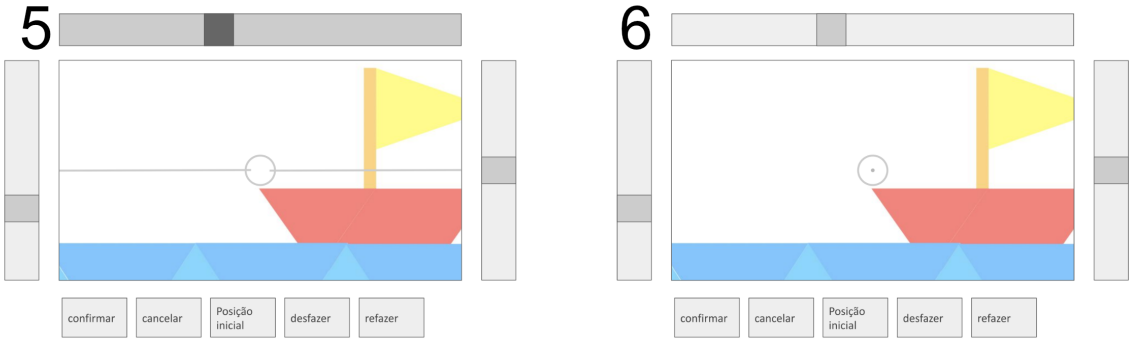


**Figura 2.5**

isso o olhar retorna à área central (6), confirmando a movimentação.

Homologamente ao processo anterior, em (7) a barra de deslocamento vertical é selecionada, sua cor é alterada e uma linha vertical percorre a tela. Ao mover o olhar para cima (8), a imagem é deslocada de acordo.

O olhar então retorna à área central da tela (9), finalizando a movimentação. Um pequeno ponto aparece no interior do círculo central, simbolizado as coordenadas selecionadas. Neste momento, seria possível retornar o olhar à qualquer uma das barras de seleção para alterar a posição do ponto, ou o zoom da tela. Em (10) a cor do botão "confirmar"é alterada para indicar que ele foi selecionado.

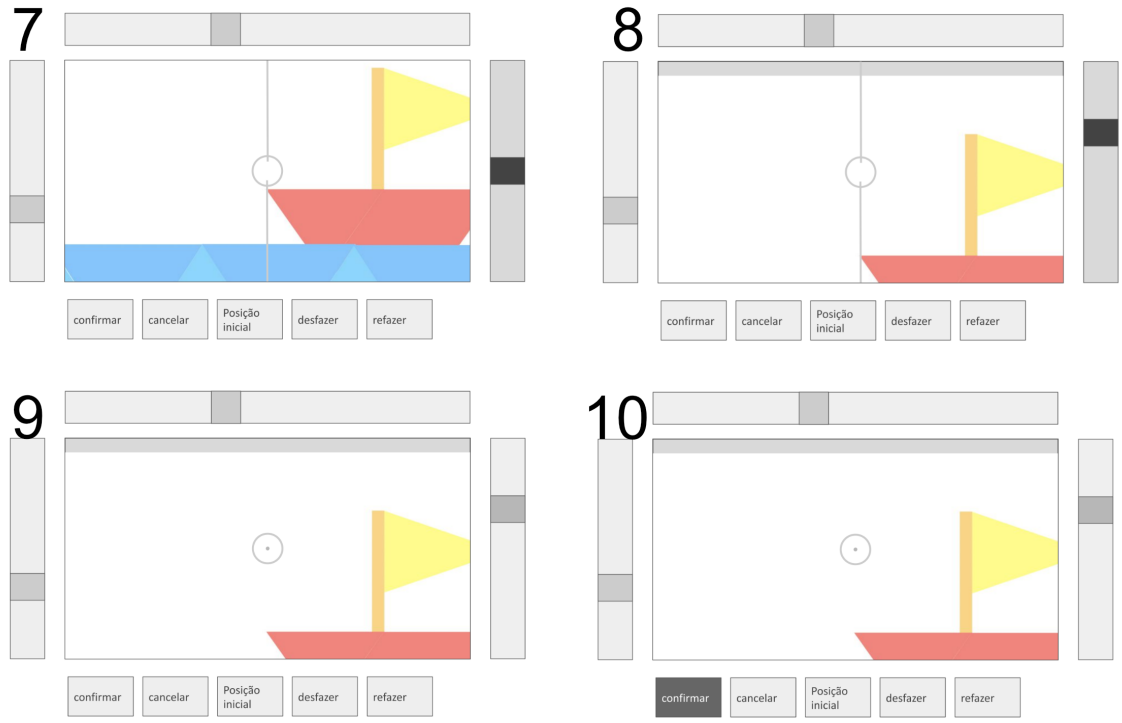


**Figura 2.6**

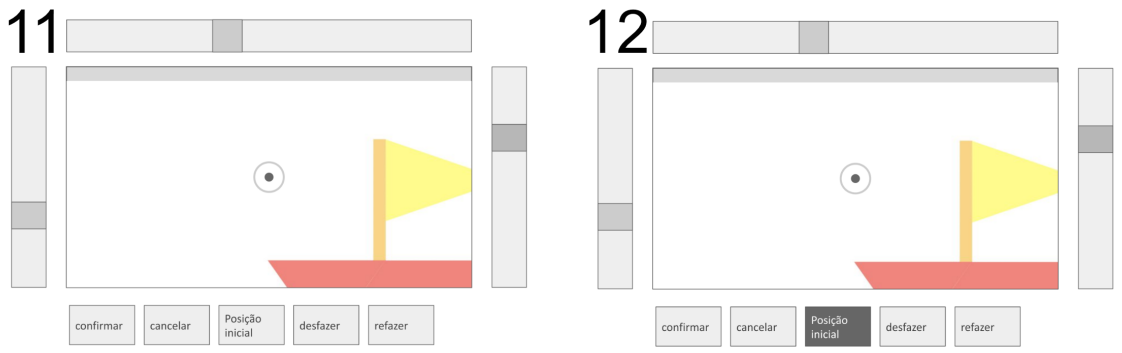


**Figura 2.7**

Um ponto definitivo foi criado (11). Isso é representado pela mudança de cor e tamanho do mesmo. Para retornar ao zoom mais distante, seleciona-se o botão "posição inicial"(12).

Em (13), o olhar retorna à ilustração central, confirmando a seleção do botão. O sistema retorna a seu estado inicial, sendo possível visualizar o ponto que foi criado.

Por meio de tal processo, é possível definir uma série de pontos que eventualmente podem formar uma imagem complexa.

Uma grande limitação identificada no sistema representado é que ele requer que o usuário alterne seu foco entre a ilustração e as barras de movimentação com grande frequência para verificar exatamente qual ponto está sendo selecionado. Uma solução,

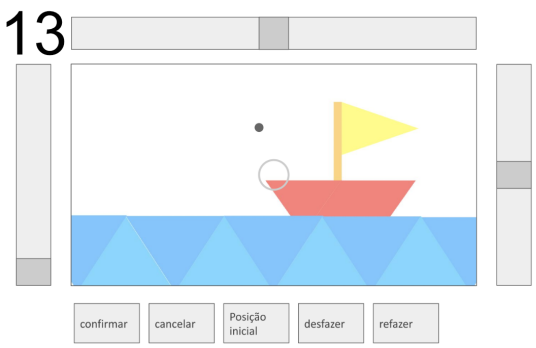


**Figura 2.8**

representada em (7a), seria representar um corte da imagem na respectiva barra quando ela estiver selecionada.

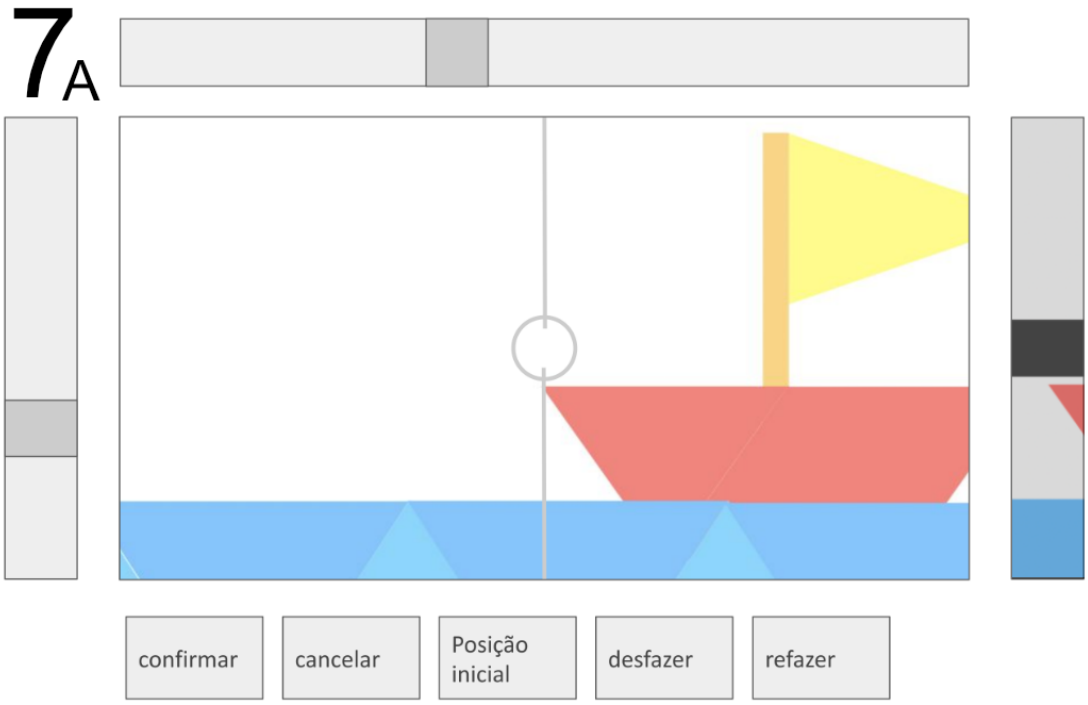


**Figura 2.9**

Em (7a) nota-se um corte do céu, barco e mar representados na barra de deslocamento vertical, à direita. Desta forma, o usuário seria capaz de visualizar o que está selecionado ao mesmo tempo que realiza a seleção, o que reduziria a fricção da interação.

# Capítulo 3

# Dotgrid: a ferramenta utilizada

Uma aplicação open source de desenho vetorial foi utilizada como base para o desenvolvimento da prova de conceito do projeto. A escolha da mesma se baseou em dois critérios: as plataformas para as quais ela foi desenvolvida e a existência de recursos de interesse considerados como básicos: a possibilidade de se definir pontos, retas e curvas de Bezier e a existêncua de figuras geométricas predefinidas, como retângulos e círculos, para agilizar o processo de criação de formas comuns.

Quatro ferramentas de open source foram consideradas, e estão resumidas na tabela abaixo.

| **Nome da aplicação** | **Criação de pontos, retas e curvas de Bezier** | **Criação de formas geométricas predefinidas** | **Plataforma** |
|---|---|---|---|
| Dotgrid | sim, com limitações | não | Web |
| Inkscape | sim | sim | Linux, Windows, OSX |
| Karbon (Calligra) | sim | sim | Linux, Windows, OSX, FreeBSD |
| Synfig | sim | sim | Linux, Windows, OSX |

Optou-se por utilizar a ferramenta Dotgrid, um software para a criação de ícones e logos. A única opção de desenho que ele oferece é a criação de pontos, a partir dos quais é possível definir retas e curvas de Bezier. Apesar de simples, ele possui os fundamentos e recursos básicos essenciais para o desenvolvimento da ferramenta. Além disso, é de fácil distribuição, visto que sua implementação é baseada em Javascript e HTML.

A interface do Dotgrid, apresentada a seguir, possui uma quantidade finita de pontos que podem ser utilizados. Dois ou mais pontos podem ser conectados por uma reta ou um arco. Uma curva de Bézier é definida por três ou mais pontos, sempre de total ímpar. Pontos ímpares pertencem à curvatura, enquanto os pares atuam como pontos de controle que definem o formato da mesma. A ilustração a seguir mostra três formas, unidas por arcos.

É possível alterar a posição de pontos e deletá-los. Curvas fechadas podem ser preenchidas por uma cor. Outras opções de personalização relevantes são a alteração da grossura

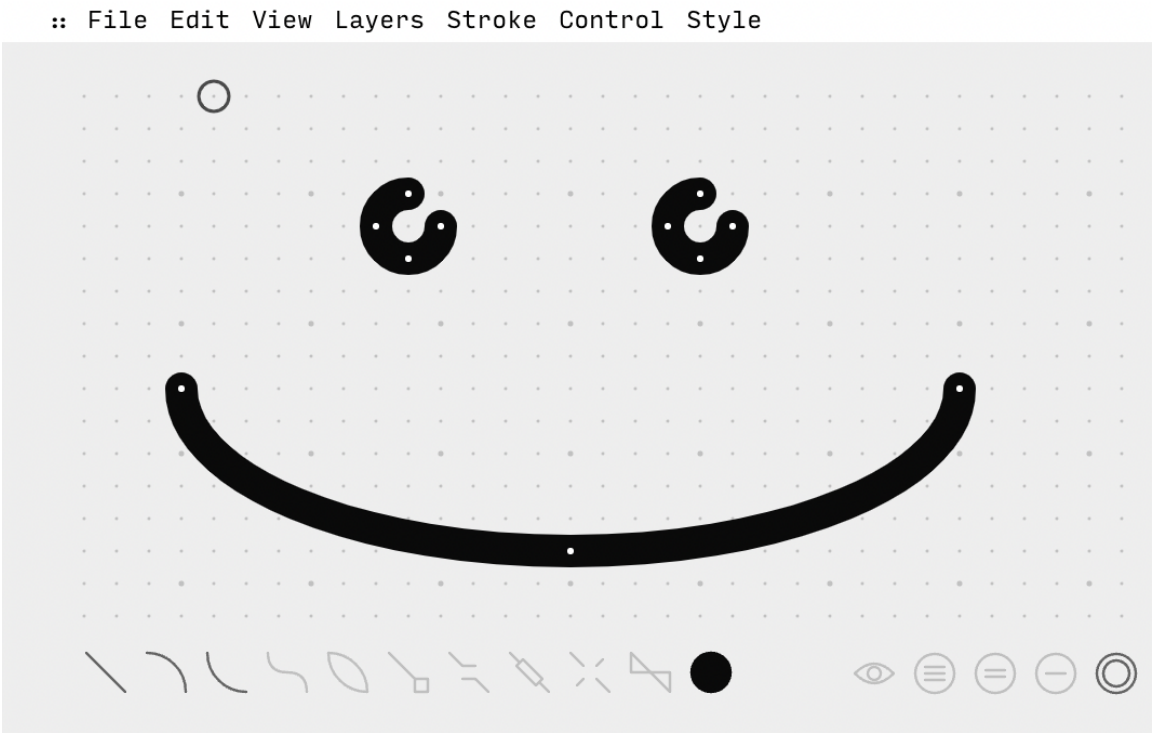


**Figura 3.1:** *Interface do Dotgrid. A aplicação web está representado três curvas geradas.*

do traço e sua cor. Não é possível criar traços de cores distintas; tais alterações se aplicam à totalidade da imagem gerada. O software permite a alteração entre três camadas, e a exportação da imagem gerada. (*Dotgrid by Rek & Devine* 2022)

O design minimalista da ferramenta indica que o foco principal é em usuários experientes, familiares com os ícones e que possam fazer uso dos diversos atalhos de teclado oferecidos. O programa foi escrito em Javascript, HTML e CSS.

## 3.1 Limitações

O Dotgrid só permite a seleção de alguns pontos predefinidos, representados em cinza ao fundo da imagem. Idealmente, deveria ser possível selecionar qualquer ponto da tela. A principal limitação, entretanto, é a inexistência de uma ferramenta de zoom da ilustração. Tal ponto não foi visto como um empecilho no momento da decisão, porém se provou um obstáculo ao decorrer do trabalho.

# Capítulo 4

# Implementação

Esta seção entrará em detalhes sobre como foi feita a abordagem à implementação da prova de conceito, assim como as dificuldades encontradas e os resultados obtidos. O código está disponível em https://github.com/talktonight/Dotgrid.

## 4.1 Familiarização com o código fonte

O código fonte do Dotgrid está disponível no GitHub. Ele consiste de um único arquivo híbridro de HTML e Javascript, index.html, com cerca de 1600 linhas de código, e dois arquivos auxiliares de CSS. O código é conciso e escrito de forma relativamente clara. Entretanto, não existe qualquer documentação do mesmo.

Não foi realizada uma tentativa de segmentar o código, o que posteriormente se provou um problema. Os desenvolvedores oferecem parte dos scripts que compõe index.html como arquivos distintos. Idealmente, index.html deveria importar tais scripts e possuir um código sucinto.

```
function Interface (client) {
  this.el = document.createElement('div')
  this.el.id = 'interface'
  this.el.appendChild(this.menu_el = document.createElement('div'))
  this.menu_el.id = 'menu'
  this.isVisible = true
  this.zoom = false
  const options = {
    cast: {
      line: { key: 'A', icon: 'M60,60 L240,240' },
      arc_c: { key: 'S', icon: 'M60,60 A180,180 0 0,1 240,240' },
      arc_r: { key: 'D', icon: 'M60,60 A180,180 0 0,0 240,240' },
      bezier: { key: 'F', icon: 'M60,60 Q60,150 150,150 Q240,150 240,240' },
      close: { key: 'Z', icon: 'M60,60 A180,180 0 0,1 240,240  M60,60 A180,180 0 0,0 240,240' }
    },
```

**Figura 4.1:** *Excerto do código fonte original, disponível do Github do projeto. Nele, estão representadas a criação de uma nova div, assim como a criação de ícones por meio da definição de suas coordenadas.*

O processo de tentar remover os trechos de tais scripts de index.html se provou mais complexo do que o esperado. Como ilustrado no excerto de código acima, mesmo os elementos de HTML da aplicação são gerados via Javascript. O mesmo vale para todos os elementos gráficos, como as linhas e curvas que ilustram os botões.

## 4.2 Prova de conceito

Inúmeras alterações foram realizadas ao Javascript e CSS para tornar a aparência da ferramenta compatível com aquela dos protótipos de baixa fidelidade. O aspecto final da ferramenta se encontra na imagem abaixo. Assim como nos protótipos ilustrativos, a barra esquerda representa a seleção do zoom. A superior e da direita seriam respectivamente responsáveis pela translação vertical e horizontal da imagem. O círculo central representa o ponto a ser selecionado.

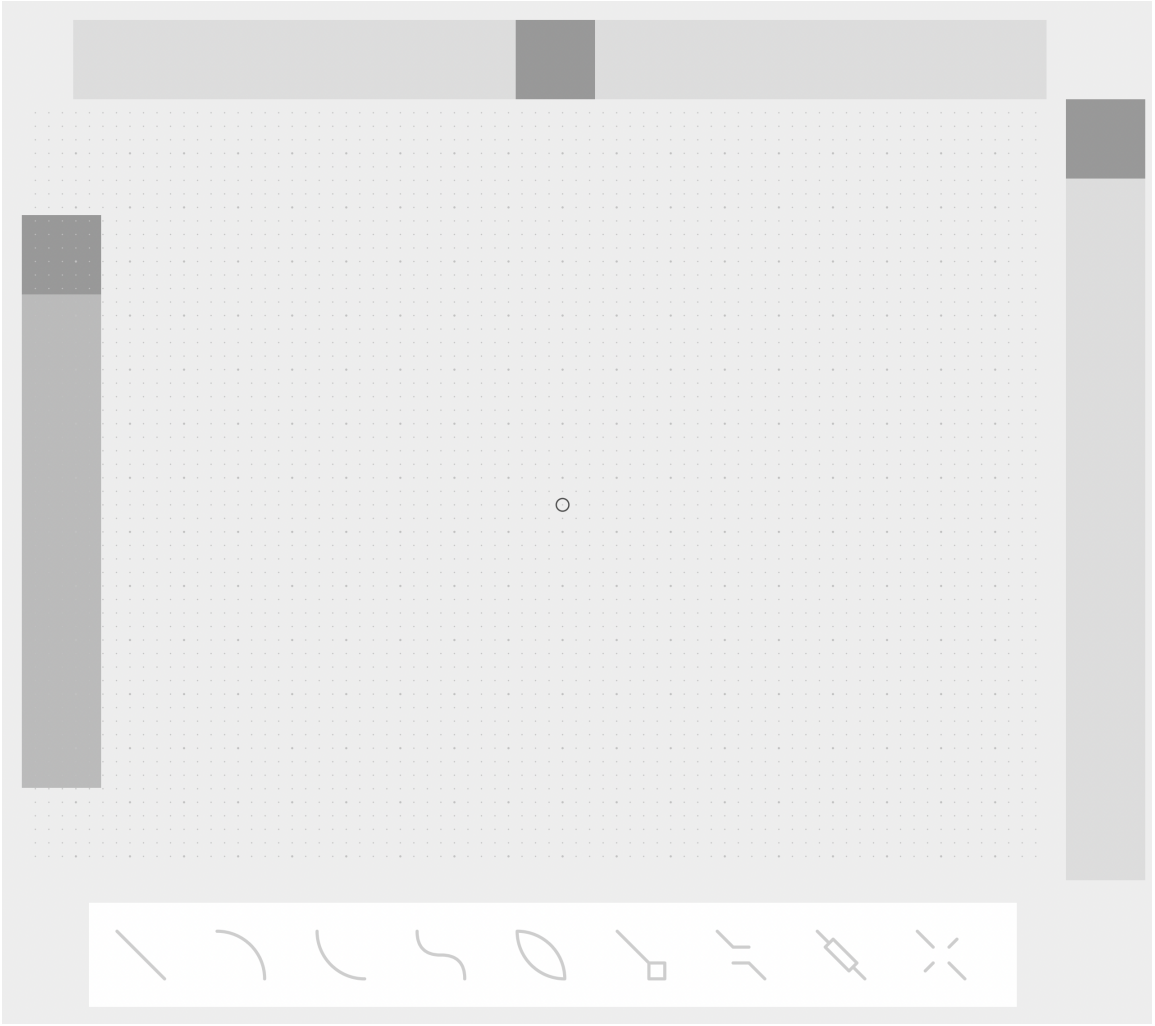

**Figura 4.2:** *Interface final da prova de conceito criada.*

A funcionalidade do protótipo é limitada. A rolagem das barras não resulta na translação da imagem ou no aumento do zoom. Ademais, as barras da esquerda e da direita apresentam comportamento irregular, nem sempre sendo clicáveis. Optou-se por uma cor distinta para

a barra da esquerda, referente ao zoom, de forma a indicar que ela deveria apresentar um comportamento distinto das outras duas.

**Figura 4.3:** *Detalhe do menu original do Dotgrid: os ícones são excessivamente pequenos para serem compatíveis com o uso de um eyetracker.*

O menu original do Dotgrid, ilustrado acima, é extremamente pequeno e minimalista. Ele foi alterado de forma a poder atuar como um GazeBar. Idealmente, o ele não deveria apresentar mais de 7 opções. Quanto maior o valor de botões disponíveis nele, maior deve ser a tela utilizada para operar a aplicação. 9 opções estão presente na ilustração presente, porém mais podem ser visualizadas caso a janela seja extendida horizontalmente.

Visando a acessibilidade, as opções deveriam ser agrupadas em submenus, que seriam apresentados acima da barra de rolavem do topo e à direita da barra de rolagem da direita. Idealmente, todas as opções de conexão entre dois pontos (reta, arco côncavo, arco convexo e curva de Bezier) em um único submenu.

Há um limite da precisão de uma seleção realizada com o olhar. A precisão de *eye trackers* é da ordem de um grau de ângulo visual (Morimoto, Leyva e Diaz-Tula, 2018). Tal valor em pixels depende da distância do usuário à tela utilizada, do tamanho da tela e a resolução em pixels da mesma. Para o cálculo do valor em pixels de um grau, utilizou-se a ferramenta Degree-to-pixel fixation radius converter, de Tomek D Loboda. Os parâmetros considerados foi uma distância do monitor de 600mm, um diâmetro de tela de 24 polegadas, resolução horizontal e vertical da tela de 4480 e 2520 pixels, respectivamente, e 1 grau de raio de fixação. Para tanto, obteve-se o resultado de que cada botão deveria ter 88.31 pixels, valor arredondado para 88 na implementação.

Todas as caixas de seleção, assim como as barras de seleção, são compatíveis com esse tamanho. Um botão com menos de 88 pixels, nas condições acima mencionadas, poderia resultar em leituras errôneas do *eye tracker*.

A remoção de algumas opções do menu que não são utilizadas também seria uma opção interessante. Idealmente, deve-se construir uma ferramenta enxuta, com poucas funcionalidades, mas que sejam facilmente operadas.

### 4.2.1 Dificuldades encontradas

A implementação do zoom se provou mais complexa do que inicialmente imaginado. O aumento do *aspect ratio* dp *canva* do JavaScript teve consequências inesperadas em outros aspectos da funcionalidade da aplicação, como na identificação de qual ponto está sendo

selecionado. A ausência de documentação do código também dificultou o processo como um todo.

Permitir a seleção de todos os pontos, ao invés de apenas alguns predefinidos, também foi uma grande dificuldade que não pode ser superada. O GazeBar é responsível ao mouse, e teoricamente poderia ser operado por um *eye tracker*. Entretanto, o mesmo não se aplica à seleção de pontos. Não foi possível implementar a confirmação de seleção de um ponto. Apesar da barra de translação horizontal, localizada no topo da tela, ser funcional, as tentativas de implementação da translação da tela não tiveram sucesso.

### 4.2.2 Ponderações finais

De início, a simplicidade e aparente clareza inicial do código do Dotgrid foram vistos como grandes pontos a seu favor. Entretanto, a inexistência de documentação e a limitação de funcionalidades resultaram em um excessivo dispêndio de tempo tentando solucionar questões "triviais". Todas as outras ferramentas consideradas como base para a aplicação apresentam um zoom implementado, por exemplo.

Um dos princípios nos quais se insiste ao se discursar sobre prototipagem é que, quanto mais tempo se dispende desenvolvendo um protótipo, mais difícil é abandoná-lo por outro — mesmo quando ele não atinge o objetivo desejado. A troca do Dotgrid por outra ferramenta de desenho vetorial Web melhor documentada, e que possua a função de zoom, teria sido a abordagem ideal.

# Capítulo 5

# Conclusão

Neste trabalho foi desenvolvido um estudo de design de interação para o desenvolvimento de uma ferramenta de desenho vetorial controlada pelo olhar. Tal ferramenta tornaria a realização de desenhos mais acessível. Mais do que isso, a intenção era que a realização da tarefa fosse o menos custosa possível. Para tanto, optou-se por fazer uso do GazeBar: um método de interação com o olhar baseado no uso de menus hierárquicos nos quais a seleção é feita por menos de troca de contexto do olhar.

Para garantir o mínimo de atrito ao interagir com o programa, métodos de design centrado no usuário foram utilizados desde as etapas iniciais do projeto. Foi possível entender como abordagens de Interação Homem-Computador podem ser aplicadas em prática para resolver um problema real. As diversas iterações de prototipação permitiram a identificação de necessidades do sistema, assim como deixaram claro que certas soluções que aparentavam ser promissoras não o eram.

A implementação da ferramenta não foi tão simples quanto o imaginado. A ausência de documentação do Dotgrid, uma ferramenta web de desenho vetorial, dificultou imensamente o processo. A insistência no uso do Dotgrid, ao invés de buscar outra alternativa, foi um erro de decisão de projeto que poderia ter sido evitado. Ademais, a funcionalidadede zoom deveria ter sido tratada com um recurso essencial das ferramentas consideradas como possível base para este projeto.

Muito tempo foi investido no estudo, familiarização e modificações do código do Dotgrid antes realizar as primeiras tentativas de implementar o zoom. Era importante ter mantido em mente quais as principais funcionalidades a serem implementadas, e ter realizados testes iniciais para identificar se seria possível codificar as mesmas no tempo disponível.

Foram diversos os desafios de implementação que surgiram ao decorrer do processo, que limitaram o que foi possível atingir. Ainda assim, o projeto foi uma excelente fonte de aprendizado. A falta de documentação do código foi uma oportunidade de aprender a estudar um código Javascript escrito de uam forma com a qual não estava familiarizada. Foi possível aprender a realizar novas funções com a linguagem. Por exemplo, entender uma nova forma de gerar de vetores para ilustrações, como ilustrado na figura X.

## 5.1 Trabalhos futuros

O passo inicial é a pesquisa por outras ferramentas de desenho vetorial para a plataforma Web, e prosseguir o desenvolvimento a partir das mesmas. Caso uma não seja encontrada, outras alternativas para a implementação de zoom no Dotgrid devem ser investigadas.

Tendo a interface completa, é necessário implementar o uso do sistema de *eye tracker* para controle de input ao invés do mouse. Testes de usabilidade devem ser realizados, tanto com usuários experientes quanto novatos. Seria possível realizar um estudo de comparação entre a performance da aplicação alterada do Dotgrid operada por mudança de contexto do olhar e o EyeDraw ou EyeSketch, ferramentas de desenho controlado pelo olhar já existentes que realizam seleções por meio de *dwell time*. Desta forma, seria possível confirmar o pressuposto inicial deste projeto de que a interação por meio de mudança de contexto reduz o atrito cognitivo e permite que o usuário realize suas tarefas em estado de *flow*.

# Referências

[Alan Cooper s.d.] Alan Cooper. *The Inmates are Running the Asylum*. 1ª ed. Sams Publishing (citado na pg. 5).

[Csikszentmihalyi 2014] Mihaly Csikszentmihalyi. *Flow and the Foundations of Positive Psychology*. en. Dordrecht: Springer Netherlands, 2014. isbn: 978-94-017-9087-1 978-94-017-9088-8. doi: 10.1007/978-94-017-9088-8. url: http://link.springer.com/10.1007/978-94-017-9088-8 (acesso em 23/12/2021) (citado na pg. 2).

[Dam e Siang 2022] Rikke Friis Dam e Teo Yu Siang. *Prototyping: Learn Eight Common Methods and Best Practices*. en. url: https://www.interaction-design.org/literature/article/prototyping-learn-eight-common-methods-and-best-practices (acesso em 15/02/2022) (citado na pg. 7).

[*Dotgrid by Rek & Devine* 2022] *Dotgrid by Rek & Devine*. url: https://hundredrabbits.itch.io/dotgrid (acesso em 15/02/2022) (citado na pg. 14).

[Elmadjian e Carlos H Morimoto 2021] Carlos Elmadjian e Carlos H Morimoto. "GazeBar: Exploiting the Midas Touch in Gaze Interaction". en. Em: *Extended Abstracts of the 2021 CHI Conference on Human Factors in Computing Systems*. Yokohama Japan: ACM, mai. de 2021, pgs. 1–7. isbn: 978-1-4503-8095-9. doi: 10.1145/3411763.3451703. url: https://dl.acm.org/doi/10.1145/3411763.3451703 (acesso em 23/12/2021) (citado na pg. 2).

[Experience 2022] World Leaders in Research-Based User Experience. *UX Prototypes: Low Fidelity vs. High Fidelity*. en. url: https://www.nngroup.com/articles/ux-prototype-hi-lo-fidelity/ (acesso em 15/02/2022) (citado na pg. 7).

[Heikkilä 2013] Henna Heikkilä. "EyeSketch: a drawing application for gaze control". en. Em: *Proceedings of the 2013 Conference on Eye Tracking South Africa - ETSA '13*. Cape Town, South Africa: ACM Press, 2013, pgs. 71–74. isbn: 978-1-4503-2110-5. doi: 10.1145/2509315.2509332. url: http://dl.acm.org/citation.cfm?doid=2509315.2509332 (acesso em 15/02/2022) (citado na pg. 6).

[Hornof *et al.* s.d.] Anthony Hornof, Anna Cavender e Rob Hoselton. "EyeDraw: A System for Drawing Pictures with Eye Movements". en. Em: (), pg. 8 (citado nas pgs. 2, 6).

[Carlos H. Morimoto *et al.* 2018] Carlos H. Morimoto, Jose A. T. Leyva e Antonio Diaz-Tula. “Context switching eye typing using dynamic expanding targets”. en. Em: *Proceedings of the Workshop on Communication by Gaze Interaction*. Warsaw Poland: ACM, jun. de 2018, pgs. 1–9. ISBN: 978-1-4503-5790-6. DOI: 10.1145/3206343.3206347. URL: https://dl.acm.org/doi/10.1145/3206343.3206347 (acesso em 23/12/2021) (citado na pg. 2).