DM

Interações de Toque e de Caneta para Visualizações de Dados em Tablets

DISSERTAÇÃO DE MESTRADO

Marcelo José Gomes Ramos
MESTRADO EM ENGENHARIA INFORMÁTICA

UNIVERSIDADE da MADEIRA
A Nossa Universidade
www.uma.pt

fevereiro | 2020

Interações de Toque e de Caneta para Visualizações de Dados em Tablets

DISSERTAÇÃO DE MESTRADO

Marcelo José Gomes Ramos
MESTRADO EM ENGENHARIA INFORMÁTICA

ORIENTAÇÃO
Pedro Filipe Pereira Campos

CO-ORIENTAÇÃO
Diogo Nuno Crespo Ribeiro Cabral

UNIVERSIDADE da MADEIRA

Interações de toque e de caneta para visualizações de dados em *tablets*

Marcelo José Gomes Ramos

Constituição do júri de provas públicas:

Presidente:

- Karolina Baras, (Professora Auxiliar da Universidade da Madeira)

Arguente:

- Filipe Magno de Gouveia Quintal, (Professor Auxiliar Convidado da Universidade da Madeira)

Vogais:

- Pedro Filipe Pereira Campos, (Professor Associado com Agregação da Universidade da Madeira)
- Diogo Nuno Crespo Ribeiro Cabral, (Investigador Auxiliar do Instituto Superior Técnico)

Junho 2020

Funchal – Portugal

Resumo

Atualmente, a análise e a manipulação de dados são fundamentais para os trabalhadores do conhecimento. Assim sendo, torna-se crucial desenvolver interações para a visualização de dados, de modo a alavancar as capacidades de análise e a melhorar os fluxos de trabalho. Tais interações podem ser compostas por interações de toque e de caneta.

Esta dissertação propõe um método de interação para visualizações de dados: a interação bimanual de toque e de caneta. Para este método de interação foram desenvolvidas quatro ações. Estas ações são reconhecidas por uma aplicação que converte as ações do utilizador em reações nas visualizações de dados. Fez-se um pequeno estudo de bibliotecas de visualização, no qual avaliou-se as bibliotecas amCharts, AnyChart, D3.js, LargeVis e VTK.

Este sistema e os métodos de interação desenvolvidos foram alvo de um processo de avaliação, a partir do qual foi possível avaliar a viabilidade deste tipo de interação neste contexto. Foi possível determinar que este tipo de interação é praticamente igual em termos de análise das visualizações, mas que apresenta uma menor usabilidade e um maior tempo para realizar as tarefas, quando comparada com a interação com o rato e o teclado.

Palavras-chave: visualizações de dados, interação humano-computador, interação bimanual e interação com *tablets*.

Abstract

Nowadays, data analysis and manipulation are fundamental for data analysts and other people that analyse data at their jobs. As such, it becomes crucial to develop interactions for data visualization in such a way that it leverages the capabilities of analysis and increasing workflow. These interactions can be comprised of pen and touch based interactions.

This dissertation proposes an interaction method for information visualization: the pen and touch bimanual interaction. For this method of interaction, it was developed four actions. These actions are recognized by an application that converts the actions performed by the user into reactions on the data visualization. Furthermore, a small evaluation of data visualization libraries was conducted, in which the libraries: amCharts, AnyChart, D3.js, LargeVis and VTK were evaluated.

The developed system and its interactions underwent a user study, after which it was possible to evaluate the viability of this type of interactions in the context of information visualization. It was possible to determine that this type of interaction is practically the same in the analysis of the data visualizations but has a lower level of usability and an increased time to complete tasks when compared with mouse and keyboard interaction.

Keywords: information visualization, human-computer interaction, bimanual interaction, and interaction with tablets.

Agradecimentos

Em primeiro lugar, gostaria de agradecer aos meus pais por todo o apoio que me deram durante todo o meu percurso académico. Um muito obrigado por me terem proporcionado a oportunidade de continuar os meus estudos.

Também agradeço aos meus orientadores Doutor Diogo Cabral e Doutor Pedro Campos pela orientação dada.

Este trabalho foi parcialmente financiado pelo ITI, FCT/MCTES LARSyS (UID/EEA/50009/2013 (2015-2017)) e FCT/MCTES LARSyS (Projeto - UIDB/50009/2020).

Índice

Índice de figuras ..... **ix**

Índice de tabelas ..... **xiii**

Lista de acrónimos e de siglas ..... **xv**

1 Introdução **1**

1.1 Motivação 1

1.2 Objetivo 2

1.3 Contribuições 2

1.4 Estrutura do documento 2

2 Estado da arte **5**

2.1 Dispositivos com toque e caneta 5

2.2 Técnicas de interação através de toque e de caneta 9

2.3 Visualização de dados com interação multimodal 13

2.4 Bibliotecas de visualização de dados 16

2.4.1 amCharts 17

2.4.2 AnyChart 19

2.4.3 D3.js 19

2.4.4 LargeVis 21

2.4.5 Visualization Toolkit (VTK) 22

2.4.6 Sumário da análise de viabilidade das bibliotecas de visualização 24

3 Protótipo **25**

3.1 Arquitetura do sistema 25

3.2 *Hardware* utilizado 27

3.3 *Frameworks* utilizadas 29

3.3.1 bbTablet 29

3.3.2 Wacom Feel Multi-Touch SDK 29

3.3.3 Awesomium 30

3.3.4 openFrameworks 30

3.3.5 Mongoose 32

3.3.6 D3.js 32

3.4 Desenvolvimento do protótipo 33
3.4.1 Recolha dos dados relativos às interações de toque e de caneta 38
3.4.2 Interpretação dos dados de toque e de caneta 42
3.4.3 Renderização de visualizações de dados 43
3.5 Métodos de interação 48
3.5.1 Interações multimodais desenvolvidas 48
3.5.2 Gráfico de bolhas 50
3.5.3 Diagrama de Sankey 54
3.5.4 Gráfico de dispersão e de radar 57
4 Avaliação 63
4.1 Metodologia 63
4.2 Resultados 65
4.3 Discussão 71
5 Conclusões 75
5.1 Limitações 76
5.2 Trabalho futuro 76
6 Referências bibliográficas 79

Índice de figuras

Figura 2.1 – Póster a demonstrar o ecrã tátil desenvolvido por E. Johnson (Malvern Radar and Technology History Society e Johnson, 2016). 6
Figura 2.2 – Gráfico de área limitado por um domínio interativo (amCharts, 2020b). 18
Figura 2.3 – Diagrama de Sankey gerado pela biblioteca amCharts. 18
Figura 2.4 – Representação da população dos Estados Unidos da América, de 1970 a 1990, através de um gráfico de área empilhadas (Bostock, 2019b). 20
Figura 2.5 – Uma representação hierárquica, usando como dados a hierarquia da D3.js (Bostock, 2017). 21
Figura 2.6 – Exemplo de uma representação de dados em 3D de um modelo aerodinâmico (Kitware, Inc., 2014). 23
Figura 2.7 – Exemplo de uma representação de dados através de um gráfico de caixas de bigodes. 23
Figura 3.1 – Diagrama de arquitetura de sistema do protótipo desenvolvido. 26
Figura 3.2 – *Tablet* gráfico Wacom Cintiq 13HD Touch (Wacom Co., Ltd., 2015b). 28
Figura 3.3 – Wacom Pro Pen disponibilizada com o *tablet* Wacom Cintiq. 28
Figura 3.4 – Exemplo de um fluxograma da execução de um programa desenvolvido usando o openFrameworks. 31
Figura 3.5 – Representação de dados usada para testar o funcionamento da biblioteca D3.js. 34
Figura 3.6 – Dados reportados pelo *tablet* e que foram recebidos pelo computador. 36
Figura 3.7 – Teste da interligação de leitura e de interpretação de dados com a representação de dados no ecrã do *tablet* gráfico. 37
Figura 3.8 – Dados disponibilizados pela classe de acesso às API. 41
Figura 3.9 – Referencial cartesiano usado no *tablet*. 43
Figura 3.10 – Listagem de diretorias do servidor HTTP criado usando o Mongoose. 45
Figura 3.11 – Exemplo de uma página contendo uma representação de dados do tipo gráfico de bolhas. 46
Figura 3.12 – Representação do processo de renderização. 47

Figura 3.13 – Interação monomanual de seleção. 49
Figura 3.14 – Interações bimanuais desenvolvidas: a) interação com o sistema usando um dedo e variando a pressão da caneta; b) interação usando dois dedos e variando a pressão; c) interação que envolve três dedos a premir o ecrã e a inclinação da caneta num dos eixos *xy*. 50
Figura 3.15 – Gráfico de bolhas representando a hierarquia de classes da D3.js. 51
Figura 3.16 – *Pop-up* que aparece quando o utilizador realiza a ação de clicar sobre o círculo "Transitioner". 52
Figura 3.17 – Destaque das classes com uma quantidade de linhas superior à selecionada. Quantidade de pressão a ser exercida na ponta da caneta: a) menos pressão; b) mais pressão. 53
Figura 3.18 – Destaque das classes com uma quantidade de linhas inferior à selecionada. Quantidade de pressão a ser exercida na ponta da caneta: a) menos pressão; b) mais pressão. 53
Figura 3.19 – Diagrama de Sankey. 55
Figura 3.20 – Seleção de nós: a) dois nós não selecionados; b) nó "Solid" selecionado. 55
Figura 3.21 – Ligações de saída do nó "Solid" destacadas. 56
Figura 3.22 – Ligações de entrada do nó "Solid" destacadas. 57
Figura 3.23 – Reposição, para o estado original, do diagrama de Sankey: a) com várias ligações destacadas; b) após ter sido feita a ação para este ficar sem qualquer item destacado. 57
Figura 3.24 – Os gráficos de dispersão e de radar como são apresentados ao utilizador. 58
Figura 3.25 – Vista aproximada do gráfico de dispersão. 58
Figura 3.26 – *Pop-up* que aparece após utilizador pressionar sobre um círculo no gráfico de dispersão: a) opção para adicionar ao gráfico de radar; b) opção para remover do gráfico de radar. 59
Figura 3.27 – Eixo *y* modificado pela ação de inclinar a caneta para cima ou para baixo. 60
Figura 3.28 – Vista aproximada do gráfico de radar, com os dados de três veículos representados. Usando uma escala de valores: a) máximos globais; b) máximos locais. 60
Figura 3.29 – Demonstração do *pop-up*: a) estado inicial do gráfico de radar; b) *pop-up* revelado através da ação de passar por cima do nó. 61
Figura 4.1 – Representação gráfica das idades dos participantes deste processo de avaliação. 66

Figura 4.2 – Gráficos circulares contendo a informação sobre os participantes: a) situação profissional; b) habilitações académicas. 66
Figura 4.3 – Grau de experiência dos participantes em relação a gráficos e visualizações de dados. 67
Figura 4.4 – Gráfico de caixa de bigodes dos resultados obtidos no SUS. 69
Figura 4.5 – Resultados obtidos na questão qualitativa do questionário de usabilidade SUS. 69
Figura 4.6 – Percentagem de respostas corretas por tarefa e o total. 70
Figura 4.7 – Tempo médio de execução de cada tarefa. 71
Figura 4.8 – Gráfico de linhas das pontuações do questionário SUS calculadas para cada participante. 72
Figura 4.9 – Gráfico de caixa de bigodes do tempo utilizado para realizar as tarefas. 74
Figura A.1 – Exemplo de uma aplicação médica do *software* VTK (Kitware, Inc., sem data). . 87
Figura A.2 – Exemplo de um gráfico de barras criado através do *software* VTK. 87

Índice de tabelas

Tabela 2.1 – Comparação de vários dispositivos modernos com suporte de toque e de caneta. 8
Tabela 2.2 – Comparação das caraterísticas das várias bibliotecas de visualização analisadas. 24
Tabela 4.1 – Resultados obtidos dos questionários SUS para cada combinação de visualização e do método de interação testado. 68
Tabela 4.2 – Resultados do SUS agrupados por tipo de visualização e por método de interação. 68
Tabela 4.3 – Percentagem de respostas certas agrupadas por tipo de visualização e por método de interação. 70
Tabela 4.4 – Tempo médio, em segundos, que os utilizadores demoraram a realizar o teste, agrupado por tipo de visualização e por método de interação. 71

Lista de acrónimos e de siglas

2D	Duas dimensões
3D	Três dimensões
API	*Application programming interface*
CRT	*Cathode-ray tube*
CSS	*Cascading style sheet*
CSV	*Comma-separated values*
D3.js	Data-Driven Documents
DLL	*Dynamic link library*
DOM	*Document object model*
FHD	*Full high-definition*
HDMI	*High-definition multimedia interface*
HTML	*Hypertext markup language*
HTTP	*Hypertext markup language*
JSON	JavaScript *object notation*
OpenGL	Open Graphics Library
PDA	*Personal digital assistant*
SDK	*Software development kit*
SUS	Escala de usabilidade do sistema
SVG	*Scalable vector graphics*
TCP	*Transmission control protocol*
USB	*Universal serial bus*
VS	Visual Studio
VTK	Visualization Toolkit
W3C	World Wide Web Consortium
Win32	Windows de 32 bits

1 Introdução

Neste capítulo será apresentada a motivação que está na base do desenvolvimento deste trabalho, sendo também indicados os objetivos deste trabalho e, no fim, encontra-se presente a estrutura deste relatório.

1.1 Motivação

Hoje em dia, com muitas decisões a serem baseadas em dados e com os dados a serem obtidos em quantidades superiores, é importante que se utilize estes dados de um modo a que se possa detetar tendências e retirar outras informações importantes. Isto para que se possa formular decisões mais bem informadas, tendo em conta os dados. Através desta área da ciência de dados, é possível reduzir problemas bastante complexos a gráficos que sejam mais simples e muito mais percetíveis do que um conjunto de dados em bruto. Devido a isso, esta área é cada vez mais uma das áreas mais importantes da ciência de dados (Mason, 2019).

No entanto, existem alguns tipos de dados que não podem ser representados em apenas um gráfico devido à quantidade de dimensões/variáveis que estes possuem. Por exemplo, num gráfico de duas dimensões é possível adicionar mais dimensões, ao atribuir um código de cores ou ao mudar o tamanho da representação desses dados no gráfico. Mas, por vezes, chega-se a um ponto em que adicionar mais dimensões ao gráfico faz com que este fique mais confuso e que seja mais difícil de extrair informação deste. Ou, noutros casos, os dados podem estar tão interligados que é dificultada a visão do gráfico, advindo daí a necessidade de, em certos casos, criar visualizações de dados interativas.

A interatividade das representações de dados, geralmente, é um dos aspetos menos focados da área de visualização de dados (Lee et al., 2012). Estas representações de dados interativas são ainda mais úteis se forem capazes de potenciar a seleção e a manipulação direta, mas estas, muitas vezes, são feitas através dos meios de interação tradicionais, isto é, do rato e do teclado, que apenas conseguem fazer manipulação indireta, para poder realizar a seleção. Na manipulação direta, é usado um método de interação em que o utilizador realize as operações diretamente sobre o gráfico (Isenberg et al., 2013). Tendo isto em conta, o meio

de interação que apresenta estas caraterísticas é o de interação com recurso a toque e a caneta, sendo que este ainda apresenta uma maior precisão, maior rapidez, é mais intuitivo (Brandl et al., 2010) e é um modo de interação mais fluido (Keefe, 2010).

1.2 Objetivo

Este trabalho tem como objetivo desenvolver uma solução que detete as entradas (*inputs)* realizadas por um utilizador num *tablet* gráfico, através de interações bimanuais, com recurso a toque e a caneta e avaliar se esta modalidade de interação é válida no contexto de interação com visualização de dados. Estas interações irão atuar sobre uma representação de dados, permitindo, assim, a exploração e a manipulação de visualizações de dados. Para este propósito, é pretendido verificar de que modo esta modalidade de interação possa vir a facilitar a capacidade de perceção e de exploração dos dados que estes apresentam.

1.3 Contribuições

Este trabalho contribui com um protótipo que integra as seguintes *frameworks:* Awesomium, Wacom Feel Multi-Touch, bbTablet, openFrameworks e Data-Driven Documents (D3.js). Para este protótipo foram criadas interações multimodais de toque e de caneta para três tipos de visualizações de dados, sendo estas: um diagrama de Sankey (Kennedy e Sankey, 1898), um gráfico de bolhas e uma visualização que combina um gráfico de dispersão e um de radar na mesma visualização. No fim do desenvolvimento deste trabalho, foi realizada uma avaliação a este sistema desenvolvido.

1.4 Estrutura do documento

Este relatório contém cinco capítulos e a estrutura deste é a seguinte:

- no primeiro capítulo deste trabalho é feita uma introdução deste, na qual são apresentados a motivação e o objetivo;
- no segundo capítulo é apresentado o estado da arte, no qual é feita uma pesquisa de informação sobre o passado e o presente dos tópicos relacionados com este trabalho;

- no terceiro capítulo é dada uma descrição do *hardware* e do *software* utilizado, da arquitetura de sistema deste protótipo, do processo de prototipagem e do método de interação;
- no quarto capítulo é descrito o processo de avaliação do protótipo e os resultados obtidos através desta avaliação;
- no quinto capítulo é apresentada uma conclusão sobre este trabalho, bem como algumas futuras considerações sobre o mesmo.

2 Estado da arte

Neste capítulo serão apresentados trabalhos relacionados com dispositivos capazes de detetar entradas de toque, de caneta ou de ambos em simultâneo, com a pesquisa na área de interação humano-computador, mais precisamente a interação através de toque e de caneta. Também são demonstrados alguns trabalhos na área de visualização de dados e algumas bibliotecas de visualização.

2.1 Dispositivos com toque e caneta

Atualmente, existem inúmeros dispositivos que possuem um ecrã tátil, quer sejam *smartphones*, *tablets*, computadores ou mesmo outros dispositivos criados com um fim específico. Apesar de os primeiros desenvolvimentos destes dispositivos surgirem na década de 1960, apenas recentemente é que começaram a chegar às massas, cerca de quatro décadas depois do seu aparecimento.

O primeiro dispositivo que pegou na ideia de usar um dedo diretamente no ecrã para interagir com as opções presentes no mesmo apareceu em 1965 com as publicações de (Johnson, 1967, 1965), que pretendia aplicar esta nova tecnologia num sistema de controlo de tráfego aéreo, com o intuito de facilitar, agilizar e reduzir o número de erros no trabalho realizado pelos controladores.

Este primeiro sistema, demonstrado na Figura 2.1, era constituído por um molde que encaixava por cima de um ecrã e que nele continha vários fios, que se encontravam expostos apenas em certas partes do ecrã, dispostos em grelha, nos quais o utilizador poderia premir, para desencadear uma ação no sistema de controlo de tráfego aéreo (Johnson, 1967, 1965; Malvern Radar and Technology History Society e Johnson, 2016).

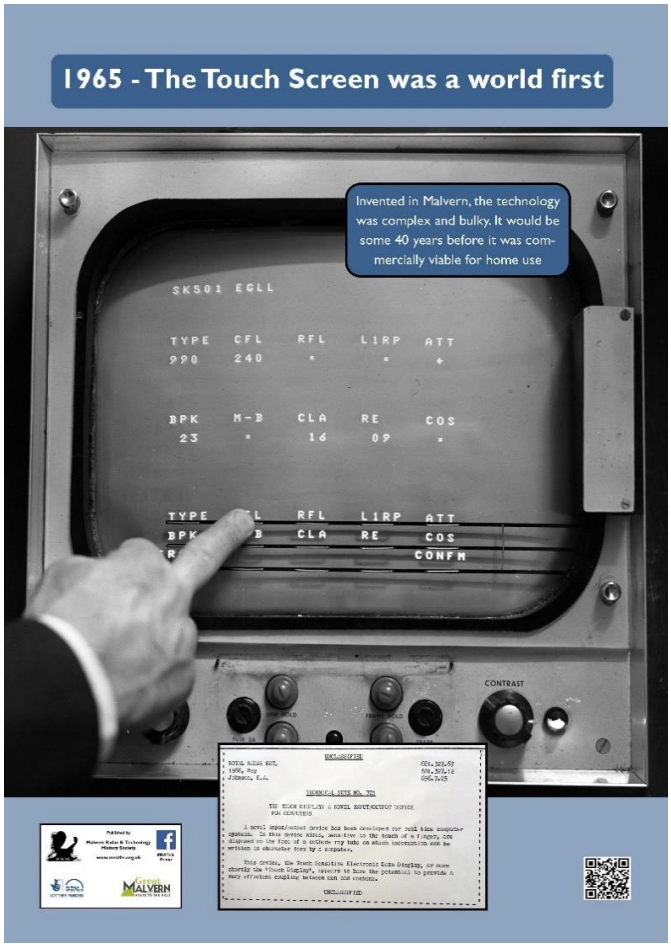


Figura 2.1 – Póster a demonstrar o ecrã tátil desenvolvido por E. Johnson (Malvern Radar and Technology History Society e Johnson, 2016).

A vantagem de um sistema destes era ser permitido que a mesma posição no ecrã possa ter várias funcionalidades nos diferentes menus e que as etiquetas, por serem apresentadas no ecrã, podem variar consoante as diferentes funcionalidades. Outra vantagem era este método de entrada ser mais rápido e mais preciso do que a utilização do teclado (Johnson, 1967).

Na década de 1970 foi quando apareceram as primeiras tecnologias em que realmente o ecrã era tátil, ao contrário do trabalho de E. Johnson (Johnson, 1967, 1965), em que o ecrã em si não era tátil. Estas duas tecnologias, que surgiram nesta década, prevaleceram ao teste do tempo e, atualmente, são as duas tecnologias mais utilizadas em interfaces de toque, sendo estas as tecnologias de ecrã tátil capacitativo e resistivo.

A primeira destas a ser desenvolvida foi a tecnologia de ecrã tátil capacitativo, em 1973 por (Beck e Stumpe, 1973), com o objetivo de aplicar este método de interação a um novo sistema de controlo de um acelerador de partículas. Este tipo de ecrã mede a diferença de

capacitância, provocada pelo dedo ao premir a superfície do ecrã, o que leva a um aumento da capacitância presente na zona de deteção do ecrã. Esta mudança é, posteriormente, detetada por um controlador que indica ao computador qual a zona do ecrã foi premida (Beck e Stumpe, 1973; Stumpe, 1977).

A segunda tecnologia a ser desenvolvida foi a de ecrã tátil resistivo, patenteada em 1975, por (Colwell Jr. e Hurst, 1975), sendo que só foi produzida na década de 1980. Esta tecnologia depende da flexão do ecrã, provocada pelo premir de um dedo na camada mais externa deste, que ao fletir entra em contacto com a camada interna do ecrã, fazendo passar corrente entre ambas. Esta passagem de corrente permite, ao controlador do ecrã, saber em que zona foi realizada uma entrada de toque.

As principais diferenças entre esta tecnologia e a de ecrã capacitativo é a necessidade de premir com alguma força, para realizar um clique neste tipo de ecrã, e a possibilidade de utilizar qualquer objeto para interagir com este tipo de ecrã (Colwell Jr. e Hurst, 1975).

É já na década de 1980, de acordo com (Buxton, 2007), que começam a aparecer ecrãs com capacidade de deteção multitoque. Um dos primeiros ecrãs a aparecer com esta tecnologia foi o ecrã desenvolvido por B. Boie nos laboratórios Bell Labs. Esta solução era composta por um ecrã do tipo *cathode-ray tube* (CRT) e uma grelha capacitativa de sensores de toque disposta por cima deste. Outra solução multitoque, desenvolvida também nestes laboratórios por (Kasday, 1984), envolvia o uso de membranas que, ao serem premidas, causavam uma refração da luz emitida pelo CRT, em que essa era direcionada para os sensores óticos colocados na borda do ecrã.

O desenvolvimento das tecnologias de toque, bem como, posteriormente, a de deteção de multitoque permitiram que, na atualidade, existam dispositivos como *smartphones, tablets* e computadores que usem estas tecnologias como método principal de interação ou como complementos de interação aos métodos tradicionais de entrada. Estas tecnologias têm, atualmente, uma presença ubíqua no nosso quotidiano.

A Tabela 2.1 contém uma lista de vários dispositivos recentes que possuem como principal caraterística a capacidade de detetar entrada de toque e de caneta. Ao observar as caraterísticas secundárias destes, pode-se notar que muitos são uma solução *standalone*, não necessitando de estarem conectados a outro dispositivo para poderem ser utilizados,

enquanto apenas dois destes são soluções complementares que necessitam de outro dispositivo para poderem operar. As resoluções dos ecrãs destes dispositivos são todos da norma *full high-definition* (FHD) ou superiores. Todos eles têm a capacidade de detetar múltiplos dedos em simultâneo, bem como a capacidade de serem operados com uma caneta. Em relação à caneta, todos eles possuem a capacidade de detetar a pressão exercida na ponta da caneta, bem como o seu grau de inclinação. A principal diferença das canetas de cada um destes dispositivos encontra-se nos níveis de sensibilidade de pressão que a caneta consegue detetar, sendo que o Wacom Cintiq Pro (Wacom Co., Ltd., 2019a) e o Wacom Mobile Studio Pro (Wacom Co., Ltd., 2019b) têm o maior nível de sensibilidade de todos estes, com 8192 níveis de pressão. De seguida, tem-se o Microsoft Surface Pro 6 (Microsoft Corporation, 2019a) e o Samsung Notebook 9 Pro (Samsung, 2017) com 4096 níveis de pressão, o Wacom Cintiq 13HD Touch (Wacom Co., Ltd., 2015a) é o que tem a menor sensibilidade com 2048 níveis e, finalmente, o Apple iPad Pro (Apple, Inc., 2019a, 2019b) não disponibiliza qualquer informação sobre a sensibilidade da caneta.

Tabela 2.1 – Comparação de vários dispositivos modernos com suporte de toque e de caneta.

Caraterísticas	**Dispositivos**					
	Wacom Cintiq 13HD Touch	Wacom Cintiq Pro	Microsoft Surface Pro 6	Wacom Mobile Studio Pro	Apple iPad Pro	Samsung Notebook 9 Pro
Ano	2015	2018	2018	2017	2019	2017
Tamanho (polegadas)	13”	13”	12,3”	13	12,9”	15”
Resolução	1920 x 1080	1920 x 1080	2736 x 1824	2560 x 1440	2732 x 2048	1920 x 1080
Solução *standalone*	Não	Não	Sim	Sim	Sim	Sim
Multi-toque	Sim	Sim	Sim	Sim	Sim	Sim
Caneta	Sim	Sim	Vendida em separado	Sim	Vendida em separado	Sim
Alimentação de corrente da caneta	Campo magnético	Campo magnético	Pilha	Campo magnético	Bateria interna	Campo magnético
Deteção da pressão da caneta	Sim	Sim	Sim	Sim	Sim	Sim
Níveis de sensibilidade de pressão	2048	8192	4096	8192	Informação não disponível	4096
Deteção de inclinação da caneta	Sim	Sim	Sim	Sim	Sim	Sim
Quantidade de botões na caneta	4	4	2	4	2	3

A Wacom é a única marca, presente na avaliação, que disponibiliza funcionalidades para aceder a todos os dados sobre a interação de toque e de caneta e que estes possam ser acedidos através de alguma biblioteca, não sendo necessário programar o acesso a esta informação de raiz. Existem outras limitações como, por exemplo, no Apple iPad Pro que permite aceder à informação sobre a pressão da caneta, mas como os eventos de toque e de caneta são reportados como eventos de toque genéricos, não é possível identificar que eventos foram gerados pela caneta ou pelos dedos, impedindo, assim, que se possa usar este dispositivo num contexto de interação bimanual de toque e de caneta (Apple, Inc., 2020a, 2020b). Sendo escolhido que seria usado um dos dispositivos Wacom, então, foi escolhido o *tablet* Wacom Cintiq 13HD Touch (Wacom Co., Ltd., 2015a), porque, de todos os dispositivos Wacom presentes nesta tabela comparativa, era o único *tablet* que poderia ser disponibilizado para o desenvolvimento deste trabalho.

2.2 Técnicas de interação através de toque e de caneta

A interação através de toque e de caneta é um tipo de interação bimanual, isto é, é um tipo de interação em que ambas as mãos são utilizadas em simultâneo para realizar operações sobre o sistema. No contexto de seleção de alvos presentes no ecrã, cuja era realizada através de interações monomanuais com um rato, com toque de um dedo ou bimanual de toque com dois dedos na sua experiência, concluíram que as interações com recurso a toque são mais rápidas e precisas do que com recurso a um rato (Kin et al., 2009). A mesma conclusão foi obtida por Karat et al., ao aplicar este método de interação num contexto de navegação de menus a uma aplicação (Karat et al., 1986).

As interações bimanuais de toque conseguem ser mais rápidas, mas apresentam um rácio de falhas superior quando comparado com apenas uma mão (Kin et al., 2009), sendo que, em geral, as pessoas apresentam uma preferência pela utilização do toque ao invés dos métodos tradicionais de entrada (Karat et al., 1986). No entanto, a utilização de ambas as mãos em simultâneo também levanta alguns problemas, tal como a possibilidade de ocorrerem colisões ou a posição de uma mão interferir com a outra devido ao posicionamento de uma. Outro possível problema que pode surgir é o de uma das mãos obstruir a visão em relação ao ecrã, como evidenciado por (Yee, 2004).

Também é necessário ter em atenção que deve-se ter em consideração que, dependendo de qual seja a mão dominante de um utilizador, é possível provocar embaraço ao usar uma interface, pelo que se deve permitir a troca para um *layout* que favoreça a mão dominante de cada utilizador (Yee, 2004).

Mack e Lang (Mack e Lang, 1989) usaram um método de deteção de toque e de caneta que convertia as entradas realizadas em eventos de rato. Estes eventos foram aplicados a um sistema gráfico de janelas, desenhado para a utilização com o rato. Com os testes que foram realizados, concluíram que com sistemas rudimentares era possível imitar todas as interações de manipulação que um rato era capaz de fazer sem ocorrer uma redução da usabilidade do sistema. No entanto, observaram que ocorriam mais erros ao usar a caneta do que o rato, mas não o suficiente para reduzir a satisfação dos utilizadores com a caneta. Também puderam encontrar evidência estatística de que as tarefas eram concluídas com maior rapidez ao usar uma interface de toque.

Os autores concluíram que a utilização do dedo como elemento apontador é uma situação confrangedora para o utilizador, devido a este método apresentar um valor de erros superior, causados pela dificuldade de selecionar objetos na interface e pelos erros de seleção a escolher ficheiros que se encontrem dentro de diretorias. Esta ideia foi, posteriormente, confirmada por Mack e Montaniz (Mack e Montaniz, 1991).

Brandl et al. (Brandl et al., 2010), através de uma experiência em que desenvolveram um labirinto no qual os utilizadores tinham que navegar usando interações bimanuais com dois dedos, um de cada mão, um dedo e uma caneta ou duas canetas, concluíram que a interação bimanual com recurso a toque e a caneta era a mais precisa, mais rápida na navegação e aquela que os utilizadores acabam por ter uma maior preferência.

As interações com recurso ao toque e à caneta podem ser criadas de maneira a serem fáceis de aprender, ao ponto de se tornarem em ações quase naturais após terem sido aprendidas. Os utilizadores também facilmente descobrem os limites do sistema através de tentativa e erro e tendem a não sair fora destes limites após os descobrirem, como verificado num estudo feito por (Walny et al., 2012). Isto deve-se ao facto de os participantes transferirem conhecimento prévio do mundo físico, quando a tarefa a realizar possui uma contrapartida física direta. Outro fator para a rápida aprendizagem tem a ver com a experiência prévia que as pessoas possuem de sistemas semelhantes, visto que, atualmente,

a maioria das pessoas já teve algum contacto com dispositivos que possuam capacidades de toque, que usem uma caneta ou até mesmo ambos para interagir com este. Outro ponto importante é que as ações realizadas devem ter sempre qualquer tipo de *feedback*, nem que seja para dizer que a ação não foi reconhecida.

PanoramicData (Zgraggen et al., 2014) é uma aplicação cuja interface é baseada num modelo de interação bimanual de toque e de caneta com o intuito de permitir a exploração e a análise de dados multidimensionais, seguindo um conceito-chave para a interação: *Unbounded Space*. O conceito *Unbounded Space* implica que os utilizadores, em situações em que estejam a realizar tarefas com uma componente cognitiva pesada, não tenham a necessidade de alternar entre vistas para visualizar os componentes de um conjunto de dados, uma vez que ao fazer isto, ocorreria uma redução de desempenho da realização da tarefa. Ao providenciar um espaço de trabalho sem limitações, que inclua técnicas simples para gerir esse espaço, muitas tarefas críticas de análise podem beneficiar de uma simplificação para tarefas implícitas, através de atividades familiares.

Kim et al. (Kim et al., 2019) desenvolveram uma aplicação de criação de narrativas derivadas de dados, com o intuito de facilitar a perceção de padrões presentes nos dados, sem que seja necessário que o utilizador tenha literacia na área de estatística. Foi aplicado um sistema de *storyboarding* que junta a análise e a apresentação de dados num ambiente suportado por interação bimanual de toque e de caneta. O que conseguiram encontrar em termos de interação foi que, os utilizadores que nunca tenham tido uma experiência prévia com sistemas de interação bimanual, tendem a ter uma dificuldade acrescida em aprenderem e/ou se habituarem a este estilo de interação. Este facto contradiz, em parte, os resultados obtidos por (Walny et al., 2012), levando a crer que nem todos os utilizadores têm a capacidade de aprender rapidamente a interagir com o sistema. Por vezes, os utilizadores ficam frustrados ao usar o sistema por tentarem utilizar o toque e a caneta de forma intercambiável, o que não é permitido pelo sistema. No entanto, no fim da experiência, os utilizadores indicaram que este tipo de interação é bastante cativante e que a sua utilização permite um maior nível de criatividade, tornando o sistema desenvolvido em algo único e divertido (Kim et al., 2019).

Hamilton et al. (Hamilton et al., 2012), no contexto de aplicação de interação bimanual de toque e de caneta a um jogo de estratégia em tempo real, utilizaram um *tablet* gráfico Wacom

conjuntamente com a tecnologia de multitoque ZeroTouch, desenvolvida por (Moeller e Kerne, 2012), para aplicarem o processo de desenho baseado no conceito *Pen-in-Hand Command* (Hamilton et al., 2012).

Pen-in-Hand Command é um conceito que faz uso de ambas as mãos em simultâneo, sendo que a mão dominante pode utilizar a caneta, assim como o toque de forma intercambiada. Este conceito é uma extensão da teoria da cadeia cinemática, teorizado por (Guiard, 1987). Através deste trabalho, observou-se que muitos utilizadores usaram a mão dominante para realizar ações de manipulação em detrimento da mão não dominante, como descrito no conceito *Pen-in-Hand Command*. Este método de interação levou a que os utilizadores se sentissem mais envolvidos e entretidos com o jogo. Em termos de praticidade, este método é mais prático que usar uma combinação de rato e de teclado em controlo de mapa, de unidades e providencia uma maior consciência do que se passa no jogo. Em operações mais complexas, os utilizadores preferem a utilização do rato e do teclado, levando a crer que, para estas operações mais complexas, os utilizadores deparam-se com uma curva de aprendizagem mais íngreme, o que afeta o desempenho destes (Hamilton et al., 2012), devendo o desenvolvedor das interações bimanuais ter o cuidado de manter as mesmas o mais simples possível.

Após rever a literatura sobre técnicas de interação bimanual de toque de caneta pode-se concluir que esta é mais cativante e satisfatória para os utilizadores, quando comparada com a interação com o rato e o teclado. Sendo também mais intuitivas e de aprendizagem rápida. Desta revisão surge o conceito *Unbounded Space* que indica que a interação não deve ser constrangida pelo espaço de trabalho, isto é, deve ser possível realizar as operações em qualquer sítio do espaço de trabalho e que este deve ser fácil de gerir. Tendo este conceito em mente, a interface a ser desenvolvida deverá ser o mais simples de utilizar e também será possível interagir em qualquer zona da interface. Outro conceito que surge é o *Pen-in-Hand Command* que indica que é possível utilizar a mesma mão que está a usar a caneta para realizar a interação de toque e a de caneta, usando os dedos ao mesmo tempo que se usa a caneta.

2.3 Visualização de dados com interação multimodal

A interação para a visualização de dados é um ramo de interação humano-computador, no qual, ao invés de o foco ser sobre a interação entre o utilizador e o computador, é focada a interação do utilizador com os dados, em que o computador age como um mero intermediário entre ambos. Nesta modalidade de interação tenta-se facilitar o processo de formulação de ideias num processo de análise de dados (Dimara e Perin, 2020).

Estes autores realizaram uma revisão sistemática sobre o tópico de interação para a visualização de dados, na qual compararam vários artigos. Após compararem estes artigos, sugeriram a seguinte definição:

> *"Interação para visualização é a interação entre uma pessoa e uma interface de dados envolvendo um propósito relacionado com os dados, em que, pelo menos, uma ação da pessoa e uma reação da interface seja percebida como tal." – (Dimara e Perin, 2020).*

Por outras palavras, a interação para a visualização de dados dá-se quando se tem uma interface em que o foco da interação é a realização de operações sobre dados, por parte do utilizador, com o propósito de modificar ou apenas alterar visualmente os dados de um modo a que este possa perceber melhor os dados que lhe estão a ser apresentados na interface.

A exploração de modalidades de entrada diferentes das tradicionais, isto é, do rato e do teclado, pode permitir a utilização de interações mais avançadas e, ao mesmo tempo, reduzir o número de elementos de janelas, de ícones, de menus e de apontadores (*windows, icons*, *menus* e *pointers* – WIMP) na interface de utilizador. Esta redução pode permitir ao utilizador do sistema focar-se mais na tarefa de analisar e de explorar os dados, em vez de estar a distrair-se com a interface (Lee et al., 2012).

Baseando-se nesta noção de usar menos elementos WIMP na interface, Jo et al. (Jo et al., 2017) criaram o programa TouchPivot que combina alguns elementos WIMP com uma interface desenvolvida para ser utilizada com interação de toque ou de caneta por pessoas inexperientes. Este sistema de manipulação de dados foi posto à prova, com testes de utilizadores, contra duas ferramentas comerciais de visualização de dados que transformam dados em folhas de cálculo em visualizações em tempo real. Desta avaliação, concluíram que,

com um pouco de treino os utilizadores, mesmo que inexperientes, conseguem responder a questões sobre os dados mais rapidamente e também que tinham uma maior facilidade de criar gráficos úteis para as tarefas que lhes foram propostas.

O trabalho de (Zgraggen et al., 2014) levou ao desenvolvimento de uma aplicação de visualização de dados com três conceitos-chave para a visualização de dados, sendo estes: *Derivable Visualizations*, *Exposing Expressive Data-Operations* e *Boolean Composition.* Começando pelo conceito *Derivable Visualizations*, este indica que as visualizações devem permitir a sua derivação, ou seja, deve ser possível criar visualizações através de referenciação ou de reutilização de visualizações pré-existentes. Também permite uma redução de complexidade, visto que modificar uma visualização é mais fácil do que criá-la de raiz. O conceito *Exposing Expressive Data-Operations* sugere que as ferramentas de transformação e de derivação de dados devem estar expostas, ou seja, estas ferramentas devem estar sempre disponíveis de forma a não criar uma disrupção cognitiva para quando o utilizador precisar delas. Por fim, o *Boolean Composition* especifica que, para a criação de visualizações interligadas, a saída (*output)* de cada visualização deve ser feita de maneira a ser possível combinar com a saída de outras visualizações. Dessa forma, é possível criar resultados mais complexos, sendo apenas preciso perceber um pouco de lógica básica booleana.

O programa DataToon, desenvolvido por (Kim et al., 2019), tem a capacidade de criar histórias de banda desenhada que têm por base representações de dados. Após a realização de testes com utilizadores, a principal introspetiva retirada, em relação à visualização de dados, foi em alguns casos ser necessário uma maior abstração dos dados, para simplificar a carga cognitiva da pessoa ao visualizar estas representações de dados.

Sadana e Stasko (Sadana e Stasko, 2014) desenvolveram uma aplicação de visualização de dados com recurso a tecnologia de multitoque. Esta aplicação tem como foco principal a representação de dados através de gráficos de dispersão. A estes é adicionada uma componente de interatividade para facilitar a pesquisa e a navegação dos dados por parte dos utilizadores. Os métodos de interação desenvolvidos envolvem um método de seleção de zona com uma ferramenta de laço (*lasso tool*), *zoom* através de movimentos realizados com dois dedos (*pinch-to-zoom)* e um simples arrastar de dedos no ecrã. Os autores consideram que fica em aberto a possibilidade da extensão desta aplicação e os seus métodos de interação a outros tipos de representações de dados.

Ao analisar algumas ferramentas de visualizações de dados, (Keefe, 2010) observou que numa destas a interação através de rato e de teclado tornava-se numa distração para o utilizador durante uma tarefa complexa de análise de dados. Esta distração acontecia porque os utilizadores teriam uma carga cognitiva pesada, devido à complexidade da visualização, e a interação era tão complexa que aumentava ainda mais a carga cognitiva, distraindo o utilizador da análise da visualização. Isto deve-se ao facto de esta ferramenta de visualização de dados requerer muitos cliques e arrastamentos de rato, tal como a reorganização e o redimensionamento das janelas dessa ferramenta, enquanto a interação de caneta era mais fluida e mais adequada a tarefas de análise de dados mais complexas.

Romat el al. (Romat et al., 2019) desenvolveram um sistema que permite que as pessoas possam explorar os dados e externalizar os seus pensamentos utilizando interações de toque e de caneta. Este sistema possibilita a externalização dos pensamentos ao permitir que os utilizadores tomem apontamentos sobre as visualizações de dados. Este possui três modos de interface: uma interface de controlo denominada *inking* e duas desenvolvidas de raiz para este projeto sendo elas *prefix* e *postfix*. Na primeira interface é possível aplicar tinta em toda a parte do documento, como se tratasse de uma caneta e uma folha de papel. Na segunda, os utilizadores tomam apontamentos após escolherem o tipo de caneta que pretendem utilizar, sendo logo visível o que a ação irá afetar. Na terceira, é realizada a ação com a caneta e, posteriormente, é escolhido o tipo de caneta a usar, em que a ação só é visível após o utilizador escolher o tipo de caneta. Após uma bateria de testes com utilizadores, H. Romat et al. criaram uma interface híbrida que combina aspetos das interfaces *prefix* e *postfix* numa só, uma vez que a interface de *prefix* permite fazer várias operações de apontar dados de uma só vez e a *postfix* permite que o utilizador faça várias operações diferentes de seguida. Esta interface híbrida contém, assim, o melhor de cada uma das outras interfaces testadas.

Uma interface de visualização de dados, tendo por base um gráfico de dispersão, foi desenvolvida por (Büring et al., 2006) num *tablet* gráfico Wacom que estaria a simular uma interface de um *personal digital assistant* (PDA). Esta interface possuía dois modos de realizar *zoom*, sendo estes um método de *zoom* através de uma distorção de lente *fisheye*, que mantinha algum contexto dos elementos à volta uma vez que não os ocultava, e o outro era um simples *zoom* que não apresentava nenhum contexto sobre os elementos à volta da zona aumentada mas permitia, aos utilizadores, um maior controlo do que estes pretendiam

visualizar. A maioria dos utilizadores preferiu a interface usando a distorção, ao invés da interface de *zoom*, porque esta possibilita uma maior facilidade de navegação, como também uma maior precisão. Quando estão a usar um dispositivo com um ecrã pequeno, preferem manter algum contexto de navegação.

Da revisão de literatura sobre a interação multimodal para visualizações de dados, pode-se concluir que este tipo de interação permite que se crie interfaces em que a interface em si não atrapalhe, mas que até facilite a compreensão da visualização de dados, por parte dos utilizadores, quando comparada com uma interface de rato e de teclado. O conceito *Exposing Expressive Data-Operations* será tido em conta no desenvolvimento do trabalho, visto que as ferramentas de manipulação das visualizações devem estar sempre disponíveis. Outra consideração para o desenvolvimento do projeto é a utilização de métodos de seleção alternativos como a ferramenta de laço ou de magnificação descritos anteriormente.

2.4 Bibliotecas de visualização de dados

Neste subcapítulo serão apresentadas as bibliotecas de visualização de dados que foram analisadas para a possível utilização neste trabalho. Apesar de existirem imensas bibliotecas de visualização de dados disponíveis para esta dissertação, apenas foram consideradas cinco delas, das quais apenas uma será escolhida para ser utilizada. As cinco bibliotecas avaliadas foram: amCharts (amCharts, 2019), AnyChart (AnyChart, 2019), Data-Driven Documents (D3.js) (Bostock, 2019a), LargeVis (Tang et al., 2016) e Visualization Toolkit (VTK) (Kitware, Inc., 2019).

Para este trabalho foi estipulado que a biblioteca de visualização a ser usada deve cumprir um conjunto de critérios que, ao aplicá-los às bibliotecas mencionadas anteriormente, é possível escolher uma que seja adequada. Os critérios são os seguintes:

- a biblioteca de visualização deve ser de utilização gratuita, ou seja, a utilização desta não deve incorrer em custos ou, tendo uma versão paga, deve existir uma versão dela que seja gratuita sem um tempo limitado de avaliação;
- deve possuir uma licença de código aberto (*open source*), ou seja, que a licença da mesma permita que se possa fazer, livremente, alterações ao código;

- tem de ser possível estender as funcionalidades da biblioteca de visualização. Este critério visa a possibilidade de adicionar a capacidade de adicionar métodos de interação bimanual às representações de dados geradas pela biblioteca;
- e, por fim, deve ser possível criar vários tipos de representações através da biblioteca de visualização de dados.

Tendo estipulado os critérios para a escolha de uma biblioteca de visualização, procedeu-se ao processo de verificar se as bibliotecas de visualização cumprem com todos estes critérios. Se alguma destas cumprir com todos os critérios, então essa biblioteca será alvo de um processo de análise de viabilidade para a utilização no protótipo.

2.4.1 amCharts

A primeira biblioteca de visualização a ser avaliada foi a amCharts, que é uma biblioteca de visualização de dados desenvolvida em TypeScript na versão 4 da amCharts e em JavaScript nas versões anteriores deste. Apesar da versão mais recente ser desenvolvida em TypeScript, esta pode ser implementada em qualquer ambiente JavaScript. Esta é desenvolvida de modo a ser fácil de a adicionar a um projeto e a rapidamente ter alguma representação de dados a funcionar (amCharts, 2019).

Esta biblioteca cumpre com todos os critérios de inclusão propostos no início da secção 2.4, daí ter sido feita uma análise de viabilidade a esta. A amCharts possui uma panóplia de diferentes tipos de gráficos. Esta funciona com representações bidimensionais, possuindo também a capacidade de representar dados de uma forma que dá a ilusão de ser tridimensional. Isto é feito através de uma projeção ortogonal (amCharts, 2020). Por esta ser desenvolvida através de JavaScript ou de TypeScript, apresenta uma desvantagem, que é a necessidade de utilizar um renderizador de páginas *web* para que seja possível observar estas visualizações.

A Figura 2.2 e a Figura 2.3 são exemplos de algumas visualizações disponíveis para esta biblioteca de visualização. A Figura 2.2 mostra um gráfico de área com um delimitador aplicado na parte de cima desta visualização.

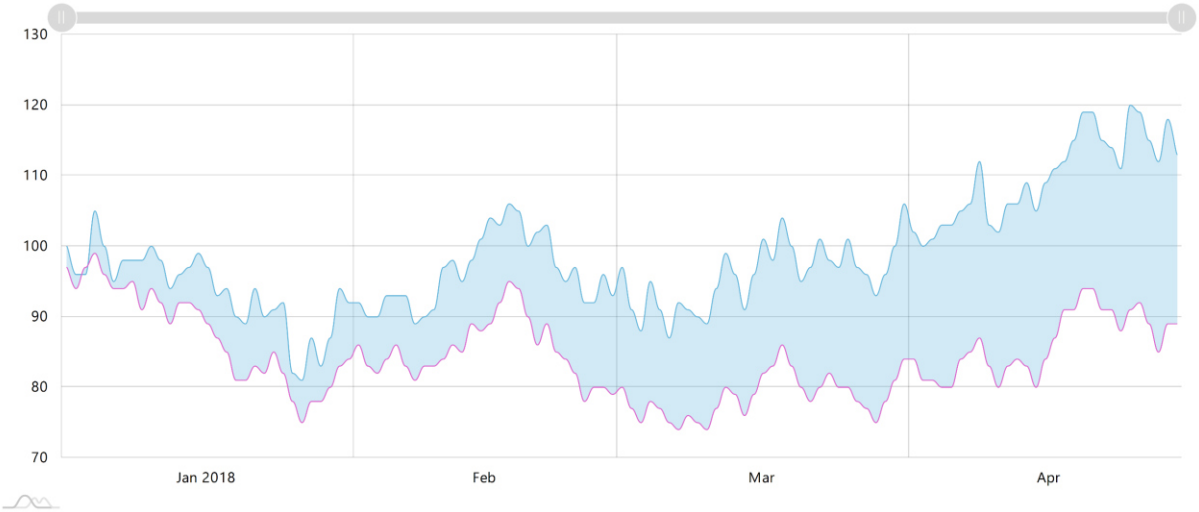


Figura 2.2 – Gráfico de área limitado por um domínio interativo (amCharts, 2020b).

Ao alterar a posição de início e de fim do delimitador, pode-se restringir os valores apresentados a um subdomínio do domínio de dados original.

Na Figura 2.3 está presente um diagrama de Sankey, no qual não existe qualquer interação predefinida.

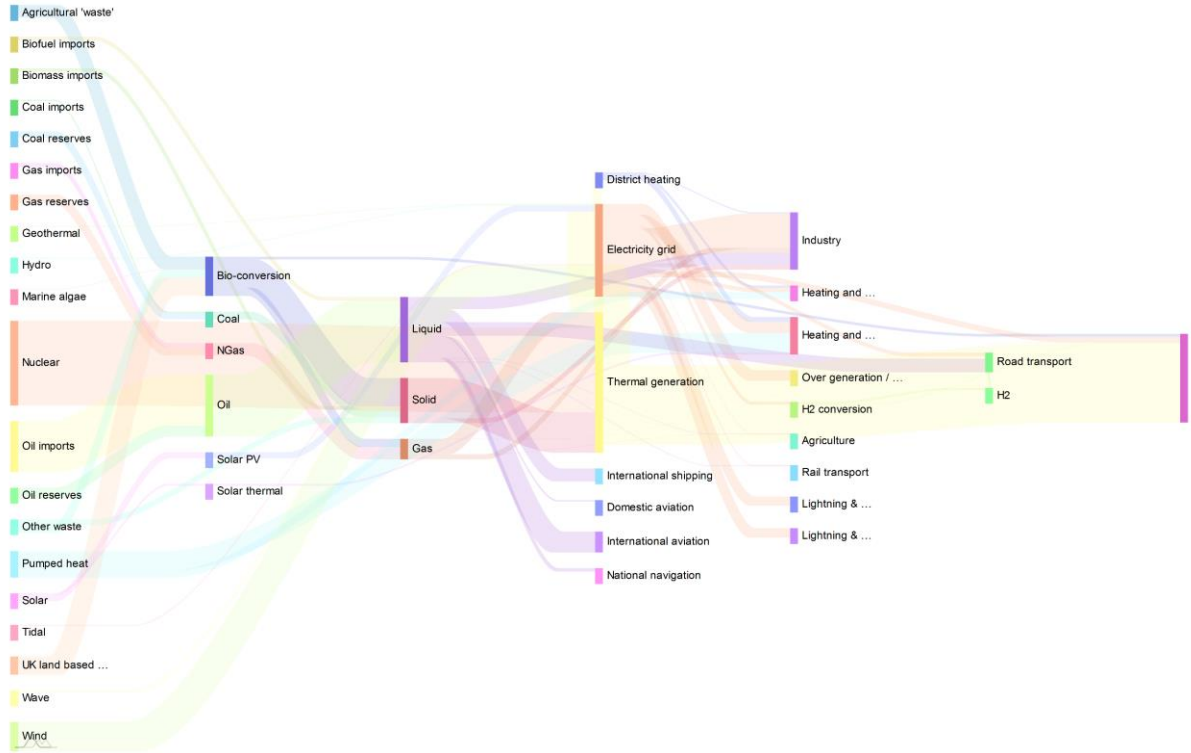


Figura 2.3 – Diagrama de Sankey gerado pela biblioteca amCharts.

2.4.2 AnyChart

Em seguida, avaliou-se a biblioteca AnyChart que é uma biblioteca de visualização baseada em JavaScript. Esta foi criada com o intuito de, principalmente, ser usada em problemas que se possam considerar como *big data*, isto é, a biblioteca AnyChart é mais usada em situações em que se tenha um conjunto de dados em grande escala (AnyChart, 2019).

Ao passar a biblioteca de visualização pelos critérios de inclusão, pôde-se verificar que esta, logo no primeiro, seria excluída uma vez que, para aceder a esta, é necessário pagar e, mesmo que tenha uma versão de acesso gratuito, esta é apenas de utilização temporária e o tempo de avaliação seria muito inferior ao tempo necessário para desenvolver o protótipo para este trabalho. Como não passou os critérios de inclusão, não foi realizada uma análise de viabilidade.

2.4.3 D3.js

A terceira a ser avaliada foi a D3.js. Esta é uma biblioteca de JavaScript desenvolvida para manipular documentos baseando-se em dados. Esta permite agregar dados a um *document object model* (DOM)[1] e aplicar transformações a este (Bostock, 2019a).

Esta biblioteca de visualização cumpriu com todos os critérios de inclusão, tendo passado por uma análise de viabilidade mais aprofundada, na qual foi possível verificar que esta possui bastantes tipos de visualizações de dados disponíveis, todas elas bidimensionais porque esta *framework* não possui a capacidade de renderização de visualizações em três dimensões (3D). No entanto, existem *plugins* que são capazes de adicionar esta capacidade à D3.js. A edição do comportamento das visualizações disponíveis é relativamente simples, porque apenas basta adicionar métodos de JavaScript para o realizar.

A Figura 2.4 e a Figura 2.5 são exemplos das capacidades da D3.js. É possível observar que esta *framework* é capaz de criar representações mais comuns, como a presente na Figura 2.4,

[1] *Document object model* ou mais conhecido como DOM é uma *application programming interface* (API) de acesso e de manipulação de documentos, particularmente HTML e XML. Através deste, é possível tornar um documento numa estrutura em árvore, na qual cada nó é um elemento presente no documento. Aos nós é possível atribuir valores e eventos programaticamente, sendo possível alterar dinamicamente o conteúdo de uma página *web* ao alterar os valores e os eventos de cada nó («DOM Standard», sem data).

ou criar umas representações mais específicas para um certo tipo de dados, como a representada na Figura 2.5.

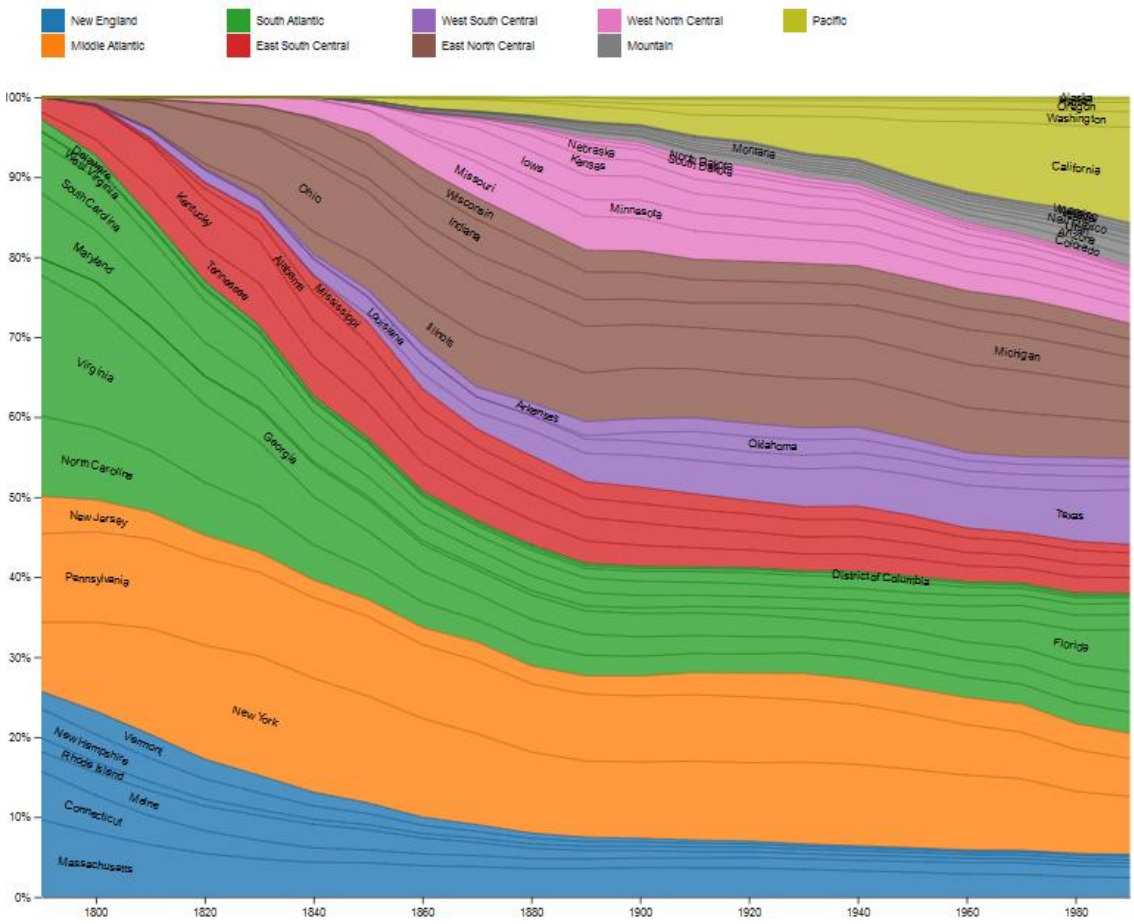


Figura 2.4 – Representação da população dos Estados Unidos da América, de 1970 a 1990, através de um gráfico de área empilhadas (Bostock, 2019b).

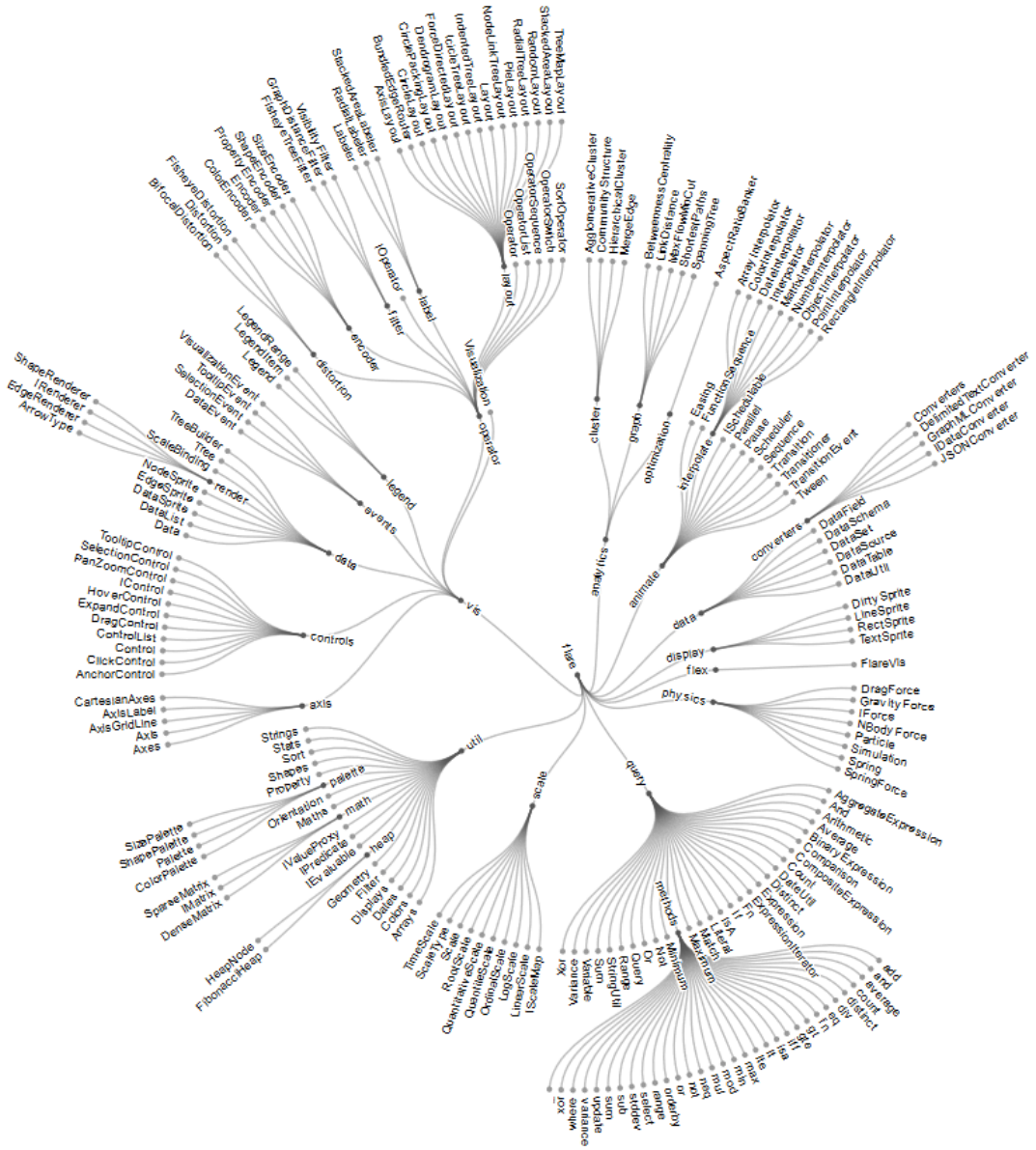


Figura 2.5 – Uma representação hierárquica, usando como dados a hierarquia da D3.js (Bostock, 2017).

2.4.4 LargeVis

Seguidamente, foi avaliada a biblioteca LargeVis, desenvolvida por (Tang et al., 2016), para visualizar problemas de grande escala ou que possuam muitas dimensões, ou seja, problemas que possuam um elevado número de variáveis o que torna muito difícil a criação de uma visualização para estes problemas. Esta funciona através do algoritmo de aprendizagem de máquina *K-nearest neighbour*, para gerar um gráfico aproximado aos dados de entrada, e depois renderiza esse gráfico como um gráfico de baixa dimensionalidade, tipicamente com duas ou três dimensões.

Esta biblioteca de visualização cumpre com os dois primeiros critérios de inclusão, mas não cumpre com o terceiro, já que esta não suporta a adição de interação às representações de dados geradas, tendo apenas a capacidade de gerar imagens estáticas. Porém, esta também seria excluída pelo facto de apenas funcionar para representar vetores de dados e redes neuronais.

2.4.5 Visualization Toolkit (VTK)

A última biblioteca de visualização a ser analisada é a VTK. Esta é desenvolvida e mantida pela empresa (Kitware, Inc., 2019). Esta biblioteca está disponibilizada em C, em C++ ou em Python. Tipicamente, ela é usada para realizar diagnósticos medicinais, simulações de dinâmicas de fluidos ou para a visualização de modelações em 3D.

Esta biblioteca de visualização é a terceira e a última em que se pôde verificar que todos os critérios de inclusão delineados foram cumpridos, logo também passou por uma análise de viabilidade. A análise desta biblioteca revelou que esta possui um enorme potencial, especialmente a nível de visualização em 3D e a manipulação das visualizações, sendo também possível a criação de modelos 3D a partir de várias camadas de imagens. No entanto, a nível de grafismo em duas dimensões (2D) esta tem menos capacidades, não tendo muitos exemplos disponíveis. Quando é preciso fazer alguma alteração para adicionar funcionalidades, começa-se a deparar com alguma dificuldade em modificar, devido à forma como esta biblioteca foi desenvolvida. A principal vantagem que esta biblioteca traria para o desenvolvimento do protótipo era que esta pode ser usada com a linguagem C++, o que permitiria uma maior facilidade em integrar a biblioteca com o resto do protótipo, porque o resto da programação seria feita nesta linguagem.

A Figura 2.6, a Figura 2.7, a Figura A.1 e a Figura A.2 apresentam exemplos do que é possível fazer ao usar a VTK. Na Figura 2.6, é possível observar um exemplo de uma representação de dados em 3D, na qual um programa faz a computação do modelo aerodinâmico de uma aeronave, que depois é representado dentro desta *framework*. A Figura A.1 representa uma visualização em que múltiplas imagens são combinadas em camadas, para formar um modelo 3D destas. Na Figura 2.7 está presente um diagrama de caixas e na Figura A.2 pode-se observar um gráfico de barras, sendo estas alguns exemplos do que se pode fazer em 2D através do *software* VTK.

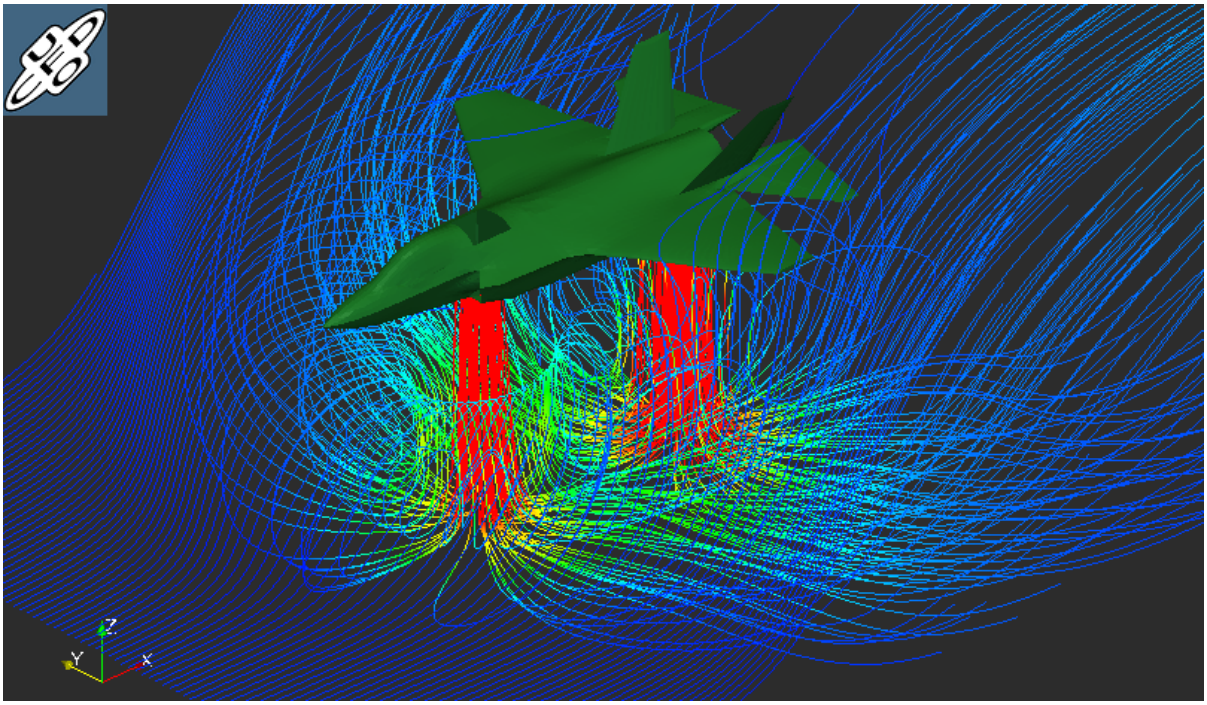


Figura 2.6 – Exemplo de uma representação de dados em 3D de um modelo aerodinâmico (Kitware, Inc., 2014).

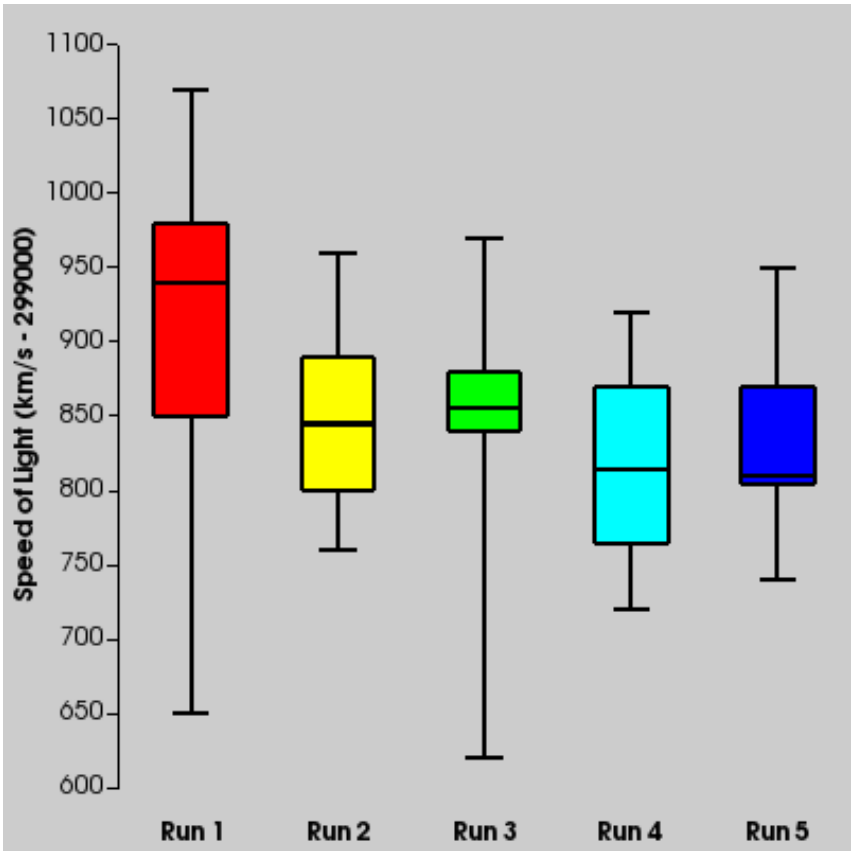


Figura 2.7 – Exemplo de uma representação de dados através de um gráfico de caixas de bigodes.

2.4.6 Sumário da análise de viabilidade das bibliotecas de visualização

Após ter-se concluído todo este processo de verificação dos critérios de inclusão a cada uma das bibliotecas e a análise de viabilidade às bibliotecas amCharts, D3.js e VTK, foi criada a Tabela 2.2. Nesta tabela estão presentes algumas das caraterísticas que foram observadas durante este processo.

Tabela 2.2 – Comparação das caraterísticas das várias bibliotecas de visualização analisadas.

Caraterísticas	Bibliotecas de visualização				
	amCharts	AnyChart	D3.js	LargeVis	VTK
Linguagem	JavaScript/Typescript	JavaScript	JavaScript	C/Python	C/C++/Python
Tipos de visualização	2D	2D	2D	2D	2D + 3D
Código aberto	Sim	Não	Sim	Sim	Sim
Acesso gratuito	Sim	Sim	Sim	Sim	Sim
Capacidade de adicionar funcionalidades	Sim	Não	Sim	Não	Sim
Complexidade para realizar alterações	Média	–	Baixa	–	Alta

O processo de escolha da biblioteca de visualização a utilizar teve em conta os dados presentes na Tabela 2.2 e algumas observações que foram feitas durante as análises de viabilidade às bibliotecas. A biblioteca VTK foi logo excluída, porque esta tem uma curva de aprendizagem muito superior às demais e seria necessário estender algumas capacidades nucleares desta biblioteca, para poder desenvolver a interação para as visualizações. A biblioteca amCharts também foi excluída, porque a programação das visualizações foi pensada para ser feita através de uma programação mais de alto nível, enquanto a D3.js opera de um modo mais baixo nível. Portanto, a D3.js permite ao desenvolvedor um maior controlo sobre a criação e a manipulação das visualizações. Para além desta diferença, a D3.js possui documentação disponível na *internet* e também um número muito maior de exemplos, quando comparada com a amCharts, sendo mais fácil de desenvolver código com biblioteca D3.js. Tendo isto em conta, a biblioteca de visualização escolhida para o desenvolvimento do protótipo foi a D3.js.

3 Protótipo

Neste capítulo será feita uma introdução da arquitetura de sistema do protótipo, do *hardware* e das *frameworks* utilizadas para o protótipo. É apresentada a evolução do protótipo na fase de prototipagem e também é apresentada uma explicação de como alguns dos módulos internos operam. No fim deste capítulo são demonstrados os métodos de interação utilizados neste trabalho, assim como as visualizações desenvolvidas.

3.1 Arquitetura do sistema

Nesta subsecção será apresentado um diagrama que corresponde à arquitetura de sistema do protótipo desenvolvido durante o curso desta dissertação. Segue-se uma breve descrição de como os vários módulos criados e as *frameworks* utilizadas se interligam.

Na Figura 3.1 observa-se que o único artefacto em que o utilizador irá realizar ações, no contexto deste protótipo, será através do *tablet* gráfico da Wacom, no qual este pode premir com os seus dedos e/ou utilizar a caneta para realizar ações no sistema. Estas ações são reportadas pelo *tablet* gráfico para o computador, através de uma conexão usando o protocolo *universal serial bus* (USB).

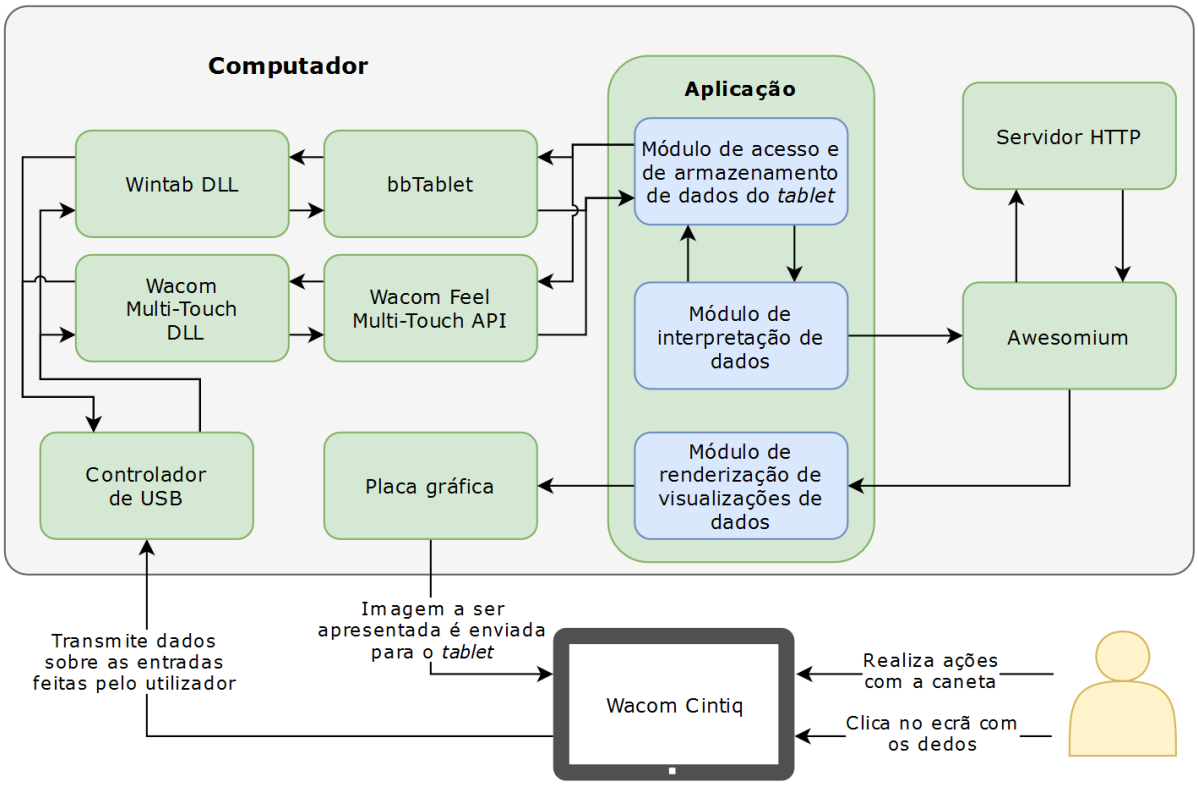


Figura 3.1 – Diagrama de arquitetura de sistema do protótipo desenvolvido.

No computador tem-se dois *dynamic link library* (DLL) e estes realizam o acesso à informação disponibilizada pelo *tablet* através do controlador de USB. Sendo estes os DLL do Wintab e o do Wacom Multi-Touch, que são utilizados pelas *application programming interfaces* (API) do bbTablet e do Wacom Feel Multi-Touch, respetivamente.

Estas API são, posteriormente, acedidas pelo módulo de acesso e de armazenamento de dados do *tablet*, sendo a API bbTablet utilizada para extrair dados da caneta e a API da Wacom utilizada para os dados sobre a deteção de dedos.

Dentro do módulo de acesso e de armazenamento de dados do *tablet*, é realizada a conversão dos dados da estrutura definida pela API bbTablet para a estrutura de dados do módulo de armazenamento de dados, sendo o mesmo feito para os dados provenientes da API Wacom Feel Multi-Touch. Este processo de conversão é mais demorado e complexo por ser necessário converter as estruturas para cada um dos dedos que possa ser detetado no ecrã. Enquanto este módulo armazena os dados, também é responsável por bloquear a leitura e/ou a escrita de dados. Este módulo é responsável por guardar os dados mais recentes, reportados pelo *tablet* gráfico, e também por enviar estes dados quando é realizado um

pedido de acesso aos dados. Enquanto está a ser feito este acesso, a escrita de dados é bloqueada.

O módulo de interpretação de dados tem como objetivo a leitura dos dados provenientes do *tablet* e depois interpretá-los. A partir dos dados, este módulo pode despoletar eventos de teclado e de rato, que são enviados para o Awesomium através do ofxAwesomiumPlus.

No módulo de renderização de visualizações de dados é feito o processo de renderização das páginas *web*, na janela da Open Graphics Library (OpenGL), gerida pelo openFrameworks. Estas páginas são geradas no Awesomium. O tratamento de mudanças de estado que, eventualmente, desencadeiam uma mudança de página *web* também é da responsabilidade deste módulo. Este comunica através do openFrameworks com a placa gráfica, de modo a que as visualizações de dados possam ser enviadas e apresentadas no *tablet* gráfico da Wacom.

Uma explicação mais detalhada dos módulos de acesso e de armazenamento de dados do *tablet*, de interpretação de dados e de renderização de visualizações de dados pode ser encontrada nas secções 3.4.1, 3.4.2 e 3.4.3, respetivamente.

3.2 *Hardware* utilizado

Nesta subsecção será apresentado o *hardware* utilizado durante o desenvolvimento do protótipo, dando uma breve descrição das capacidades, e também será explicado o propósito de cada um destes componentes no contexto do protótipo.

O *tablet* gráfico escolhido para o protótipo é o Wacom Cintiq 13HD Touch (Wacom Co., Ltd., 2015a). Este dispositivo não é um dispositivo *standalone*, o que implica que é necessário ligá-lo a um computador para que este possa funcionar. Esta ligação é feita através de um cabo *high-definition multimedia interface* (HDMI) e de um cabo USB, onde é passada, respetivamente, a informação da imagem a mostrar no ecrã e os eventos de entradas realizados pelo utilizador no *tablet* gráfico (Wacom Co., Ltd., 2015a). Este *tablet* gráfico também apresenta alguns botões no lado esquerdo do mesmo, como se pode verificar na Figura 3.2, só que, para este projeto, não lhes é atribuída qualquer funcionalidade.

Figura 3.2 – *Tablet* gráfico Wacom Cintiq 13HD Touch (Wacom Co., Ltd., 2015b).

Além deste *tablet*, vem incluída uma caneta de digitação, denominada Wacom Pro Pen, demonstrada na Figura 3.3.

Figura 3.3 – Wacom Pro Pen disponibilizada com o *tablet* Wacom Cintiq.

A caneta possui duas pontas que atuam como apontadores para o sistema operativo e, ao clicarem no ecrã, funcionam como cliques do botão principal do rato. Esta também possui outros dois botões que atuam como o botão secundário e o botão da roda de navegação do rato. Esta caneta tem um sensor na ponta mais fina, permitindo saber a pressão exercida e possuindo 1024 níveis de sensibilidade. Outro sensor que esta caneta possui é um sensor de inclinação, que consegue detetar uma inclinação de 0º a 40º nos eixos x e y (Wacom Co., Ltd., 2015a).

Para este projeto também foi utilizado um computador portátil, para poder conectar ao *tablet* supramencionado e para fazer o processamento dos dados a apresentar no ecrã e dos dados reportados pelo *tablet* sobre as entradas de toque e de caneta. Este portátil tem como principais caraterísticas um processador Intel i7-4700MQ, com quatro núcleos físicos de processamento (cada um destes é subdivido em dois subnúcleos lógicos, perfazendo um total de oito núcleos virtuais), uma placa gráfica dedicada Nvidia GeForce GT 745M e dezasseis *gigabytes* de memória interna.

3.3 *Frameworks* utilizadas

Nesta subsecção serão apresentadas as ferramentas de desenvolvimento utilizadas para desenvolver o protótipo. Para cada uma será indicado o que cada uma destas ferramentas permite fazer e o objetivo da utilização das mesmas.

3.3.1 bbTablet

Esta API é uma interface para um *tablet* de digitação, escrita em C++ por (Baxter, 2009), sendo esta compilada apenas para sistemas operativos Windows de 32 bits (Win32),mas existindo também a possibilidade de executá-la em sistemas operativos Windows de 64 bits. Esta limitação de executar apenas em sistemas baseados em Win32 tem a ver com a utilização da DLL wintab32. Este DLL é instalado no computador juntamente com os *drivers* necessários para poder utilizar o *tablet* gráfico, disponibilizados pela Wacom.

Esta API permite apenas o acesso aos dados que o *tablet* disponibiliza sobre a caneta, sendo possível saber as coordenadas cartesianas da posição da caneta no ecrã do *tablet*, no ambiente de trabalho e numa janela de um dado programa. Também é possível extrair a distância que a ponta da caneta se encontra do ecrã, a pressão exercida na ponta, o ângulo de inclinação da caneta, saber se a caneta está invertida (se tem a ponta mais grossa virada para o ecrã), se algum botão da caneta está a ser premido e, por último, se a caneta está a ser detetada pelo *tablet*.

Tanto a API bbTablet como a Wacom Feel Multi-Touch, descrita na secção 3.3.2, acedem ao *driver* wintab32. A bbTablet é utilizada para aceder aos dados sobre a caneta, em detrimento da Wacom Feel Multi-Touch, uma vez que a bbTablet faz o tratamento dos dados lidos, enquanto com a Wacom Feel Multi-Touch os dados vêm em bruto.

3.3.2 Wacom Feel Multi-Touch SDK

Esta API, desenvolvida pela Wacom (Wacom Co., 2019), permite o acesso aos dados da caneta e dos dedos, possibilitando o acesso a estes em simultâneo. O acesso aos dados relativos aos dedos foi o único componente a ser usado desta API.

Este *software development kit* (SDK) está escrito em C++ e funciona através de um sistema de *callbacks*, sendo necessário registar as funções como *callback* dos eventos que se pretende

ficar à escuta. Por exemplo, é necessário registar uma função de *callback* para receber os dados quando ocorre um evento de deteção de dedos no ecrã do *tablet*, dado esta função apenas ser executada quando for detetado, pelo menos, um dedo no ecrã; caso contrário, ela nunca será executada.

3.3.3 Awesomium

Awesomium (Awesomium e Simmons, 2017) é uma biblioteca em C++ que é capaz de adicionar um renderizador de páginas *web* a um ambiente de renderização gráfica OpenGL. Com esta biblioteca é possível criar interfaces interativas, através da utilização da *hypertext markup language* (HTML). Ao usar o navegador *web*, muitas vezes, tem-se barras de navegação e outros menus que atrapalham a interação e, ao utilizar esta biblioteca, é possível evitar o aparecimento destes, reduzindo as distrações para o utilizador.

3.3.4 openFrameworks

O openFrameworks (openFrameworks Community, 2019) é um *kit* de ferramentas de código aberto escrito em C++ e foi desenvolvido para ser uma camada que se situa hierarquicamente acima da biblioteca OpenGL. Este permite uma rápida prototipagem e experimentação, devido a ter como por base uma arquitetura de *plugins*, sendo possível encontrar muitos *plugins* que já possuem algumas funcionalidades pré-programadas, reduzindo, assim, o tempo de desenvolvimento na fase de prototipagem.

A arquitetura de *plugins* também permite uma grande extensibilidade por parte deste *kit* de ferramentas, uma vez que é possível ir adicionando funcionalidades à medida que estas vão sendo necessárias.

Uma aplicação de openFrameworks contém quatro métodos básicos de controlo da aplicação. Estes servem para inicializar o programa, atualizar o seu estado interno, renderizar o conteúdo a ser mostrado no ecrã e terminar a execução do programa. A implementação deste último é opcional, ficando a sua utilização ao critério do desenvolvedor. A inicialização só é executada uma vez, que é quando o programa é iniciado, enquanto os dois métodos de atualização e de renderização correm num ciclo e a saída da aplicação só será executada uma vez no fim da execução, como pode-se observar, esquematicamente, na Figura 3.4.

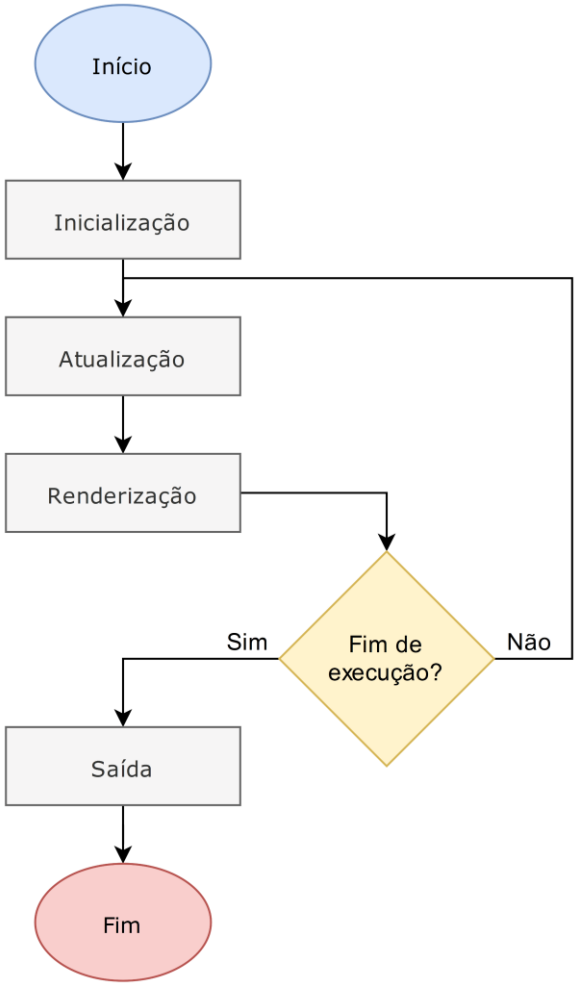


Figura 3.4 – Exemplo de um fluxograma da execução de um programa desenvolvido usando o openFrameworks.

O método de atualização é aquele em que deve ser colocado todo o código que realiza as alterações no estado interno da aplicação. O método de renderização é aquele em que é inserido o código responsável por renderizar em imagens ou em *frames* as alterações efetuadas ao estado da aplicação (openFrameworks Community, 2020).

ofxAwesomiumPlus

O ofxAwesomiumPlus (Ilin, 2016) é um *plugin* para openFrameworks que acede às funcionalidades do *kit* de ferramentas Awesomium e que permite que estas sejam utilizadas dentro de um projeto de openFrameworks. Este *plugin* é baseado noutro *plugin* denominado ofxAwesomium.

3.3.5 Mongoose

Esta biblioteca contém em si a capacidade de iniciar e de correr servidores de *transmission control protocol* (TCP), de *hypertext transfer protocol* (HTTP), de *websocket*, entre outros.

Esta biblioteca é escrita, maioritariamente, na linguagem C, sendo assim fácil a adaptação desta para um projeto de C++, requerendo apenas umas pequenas alterações de código para esta voltar a funcionar devidamente. Esta mudança de linguagem foi feita para que possa integrar todas as dependências do protótipo dentro de uma única solução dentro do Visual Studio.

O Mongoose, disponibilizado pela Cesanta Software Limited no Github (Cesanta Software Limited e Lyubka, 2019) é utilizado, neste protótipo, para criar um servidor HTTP que fará a hospedagem das visualizações de dados que serão acedidas através do openFrameworks.

3.3.6 D3.js

A biblioteca de JavaScript D3.js (Bostock, 2019a) permite a criação de visualizações de dados dinâmicas e interativas dentro de um ambiente de um navegador *web.*

Esta faz uso das tecnologias tipicamente utilizadas na criação de uma página de um *website*, tais como HTML, *scalable vector graphics* (SVG) e *cascading style sheet* (CSS), sendo estas linguagens criadas e definidas pelo World Wide Web Consortium (W3C). Quando invocada num ficheiro HTML, esta biblioteca permite a criação, a seleção, a manipulação e a estilização de elementos SVG numa página *web*, assim como a adição de transições e de efeitos dinâmicos.

Os dados a serem utilizados para a criação de visualizações de dados podem ser carregados, usando funções simples desta biblioteca, a partir de ficheiros *comma-separated values* (CSV), JavaScript *object notation* (JSON) ou geoJSON. Se houver a necessidade de carregar outros tipos de ficheiros, é possível criar funções JavaScript que façam a importação dos dados.

3.4 Desenvolvimento do protótipo

Nesta secção será apresentado como é que foi feito o processo de prototipagem, como foram realizadas certas implementações e o porquê das mesmas e algumas peculiaridades que foram encontradas durante o desenvolvimento do mesmo.

O primeiro objetivo de desenvolvimento foi o de colocar a biblioteca de renderização de visualizações de dados a funcionar e a gerar gráficos corretamente. Para poder saber qual seria a melhor biblioteca de visualização a usar neste protótipo foram criados critérios de inclusão e posteriormente foi realizada uma análise de viabilidade às bibliotecas de visualização amCharts, D3.js e VTK, como já tinha sido previamente mencionado na secção 2.4 presente no segundo capítulo. Desta análise, foi possível observar que a biblioteca de visualização de dados que era a que se melhor adequava a este trabalho era a D3.js.

A partir do momento em que a biblioteca de visualização foi escolhida, o foco foi posto em pôr esta biblioteca a correr alguns exemplos de visualizações de dados. Como esta biblioteca de visualização corre embebida em ficheiros HTML, existe a necessidade de ter um servidor para correr a mesma, sendo para esse efeito usada a biblioteca de servidores Mongoose.

Após a criação e configuração de um servidor para hospedar as visualizações de dados, foram feitos testes usando o navegador de *internet* Mozilla Firefox para poder auferir se o servidor e a biblioteca de visualização estariam a funcionar corretamente. Uma vez que este era mais rápido para testar qualquer alteração realizada, quando comparado com o navegador utilizado através do Awesomium.

Pode-se observar, na Figura 3.5, uma das representações básicas iniciais utilizadas para testar se a biblioteca de visualização de dados e se o processo de renderização das visualizações de dados se encontravam a funcionar adequadamente.

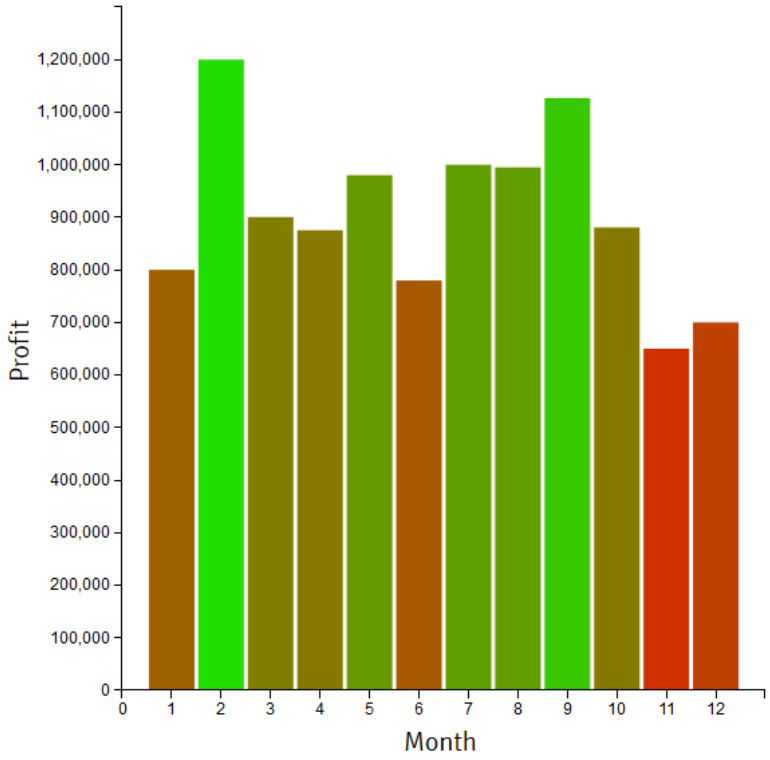


Figura 3.5 – Representação de dados usada para testar o funcionamento da biblioteca D3.js.

Nesta altura foi ligeiramente alterado o código destas visualizações, de modo a tentar perceber o que seria necessário realizar mais para a frente de modo a adicionar funcionalidades e interação às representações de dados.

Depois de colocar a biblioteca de renderização de visualizações de dados a funcionar corretamente, foi criado todo o mecanismo de acesso aos dados de toque e da caneta reportados pelo *tablet* ao sistema. Para implementar esta capacidade foi necessário instalar e configurar a biblioteca bbTablet. Processo que, inicialmente, foi moroso uma vez que é necessário transferir a biblioteca para o computador e depois é preciso fazer uma compilação desta, criando uma biblioteca estática[2] de programação com a extensão **.lib*, este ficheiro tem de ser o correto para o tipo de configuração de execução que irá ser usado no Visual Studio (VS) (Microsoft Corporation, 2019b), isto é, se na solução do VS for usado o método de compilação *debug* para uma arquitetura de 32 bits então esta tem de ser compilada com a mesma configuração. Depois de ter sido feita a configuração, começou-se a programação dos métodos de acesso à informação do *tablet*. A programação do acesso aos dados

[2] Uma biblioteca estática permite ao programador a utilização de rotinas, de funções e de variáveis externas sem que seja necessário escrevê-las de novo no projeto, reduzindo, assim, a quantidade de código reescrito ou repetido. Estas são adicionadas à aplicação alvo no momento de compilação.

disponibilizados pelo bbTablet foi feita, em parte, por um processo de tentativa e erro porque esta biblioteca possui uma documentação bastante básica no seu *website* e muitas das funcionalidades, desta API foram descobertas através de análise do código-fonte.

Após estar implementada a leitura de dados relativos à caneta, o foco passou para a implementação da deteção de dedos a premir no ecrã. Esta funcionalidade foi implementada através da Wacom Feel Multi-Touch SDK. Esta biblioteca de acesso aos dados de multitoque foi muito mais fácil de utilizar inicialmente, porque esta não necessita de ser compilada para ser usada, bastando apenas adicionar à solução de VS os ficheiros desta biblioteca. A nível de programação esta também foi mais simples, uma vez que esta possui alguma documentação no seu *website*. Com a utilização desta biblioteca foi possível descobrir que o *tablet* gráfico usado apenas consegue detetar dez dedos a pressionar o ecrã em simultâneo.

Na solução de VS foi adicionado um projeto para testar o acesso aos dados e para os poder visualizar em tempo real numa consola de comandos do Win32. Este projeto durante o desenvolvimento do protótipo serviu como uma zona de testes para programar o acesso, a leitura e a conversão de dados vindos das duas API usadas.

Na Figura 3.6 é possível observar os dados que estavam a ser extraídos do *tablet*, nesta fase de prototipagem, em relação à caneta e aos dedos a premir o ecrã e que eram apresentados na consola. Sendo esta a plataforma de testes para os dados das entradas do protótipo.

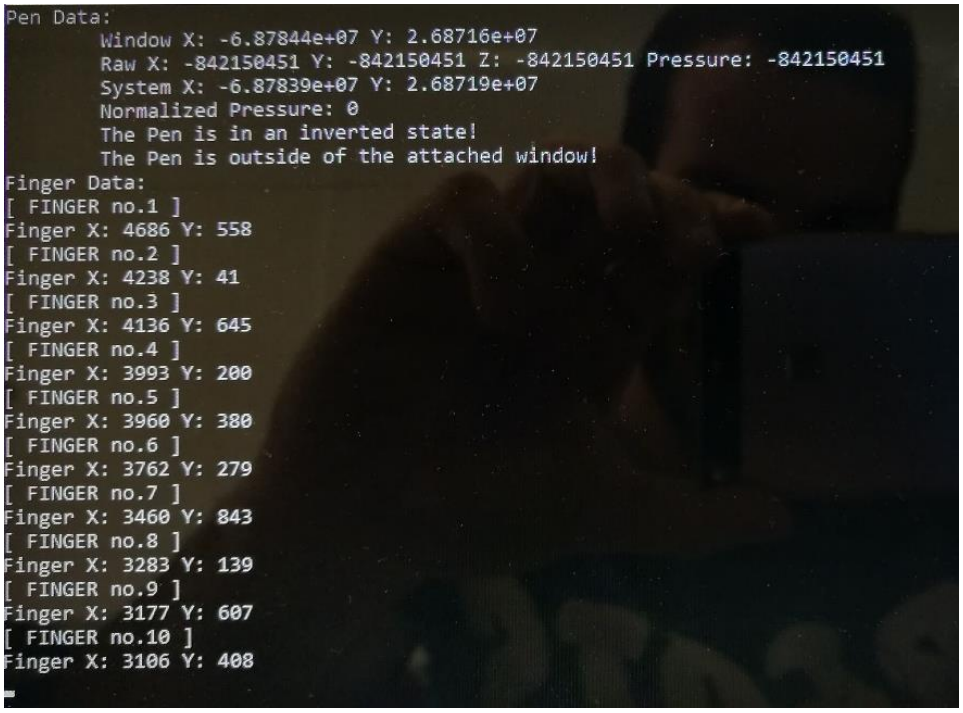


Figura 3.6 – Dados reportados pelo *tablet* e que foram recebidos pelo computador.

Após a criação de métodos para processar as entradas do *tablet* do protótipo o objetivo seguinte foi a junção destes dois projetos desenvolvidos anteriormente e verificar se a interligação das duas componentes, interação com o *tablet* e renderização das visualizações de dados, era possível. Este processo passou por juntar os dois projetos de VS, sendo estes o projeto que permite a renderização das visualizações de dados e o que realiza a leitura de dados do *tablet*, num só projeto. Estes foram unidos num só projeto, porque só assim é que seria possível enviar a informação recolhida no *tablet* para as visualizações de dados.

Ao juntar estes dois projetos deparou-se imediatamente com um problema, que surgiu devido a uma incompatibilidade. Esta deveu-se ao facto de que um dos projetos estaria a utilizar uma biblioteca estática com uma configuração para computadores de 64 bits enquanto o outro projeto estaria a usar outra que estaria configurada para sistemas de 32 bits. E, enquanto existisse esta incompatibilidade, não seria possível prosseguir com a união destes dois projetos, sendo obrigatório que ambos estivessem a utilizar a mesma, logo, para solucionar este problema, foi necessário recompilar a biblioteca do bbTablet de modo a gerar o ficheiro **.lib* adequado para este projeto. Tendo sido esta a escolhida para recompilar porque era a mais simples e mais rápida de o fazer, quando comparada com a biblioteca do openFrameworks. Este problema levou algum tempo a ser detetado porque as mensagens de

erro geradas pelo VS não são muito explícitas o que leva a que seja necessário fazer alguma pesquisa na *internet* para perceber o que a mensagem de erro significa.

Após a recompilação da biblioteca do bbTablet, começou-se a fazer uma união do código desenvolvido nos dois projetos. Este processo levou, em certas funcionalidades, a que se tivesse de fazer uma refatorização do código que sido inicialmente feito devido a incompatibilidades do acesso aos dados de toque e de caneta dentro do openFrameworks, uma vez que se teve de alterar este novo projeto de um paradigma de programação imperativo para o paradigma orientado a objetos. Depois desta refatorização e da implementação de mais alguns métodos já era possível usar o *tablet* gráfico como método de entrada para interação com as visualizações de dados. Na Figura 3.7 pode-se observar um utilizador a mexer no sistema com recurso a uma interação de caneta básica, muito similar à usada com um dedo num *smartphone*.

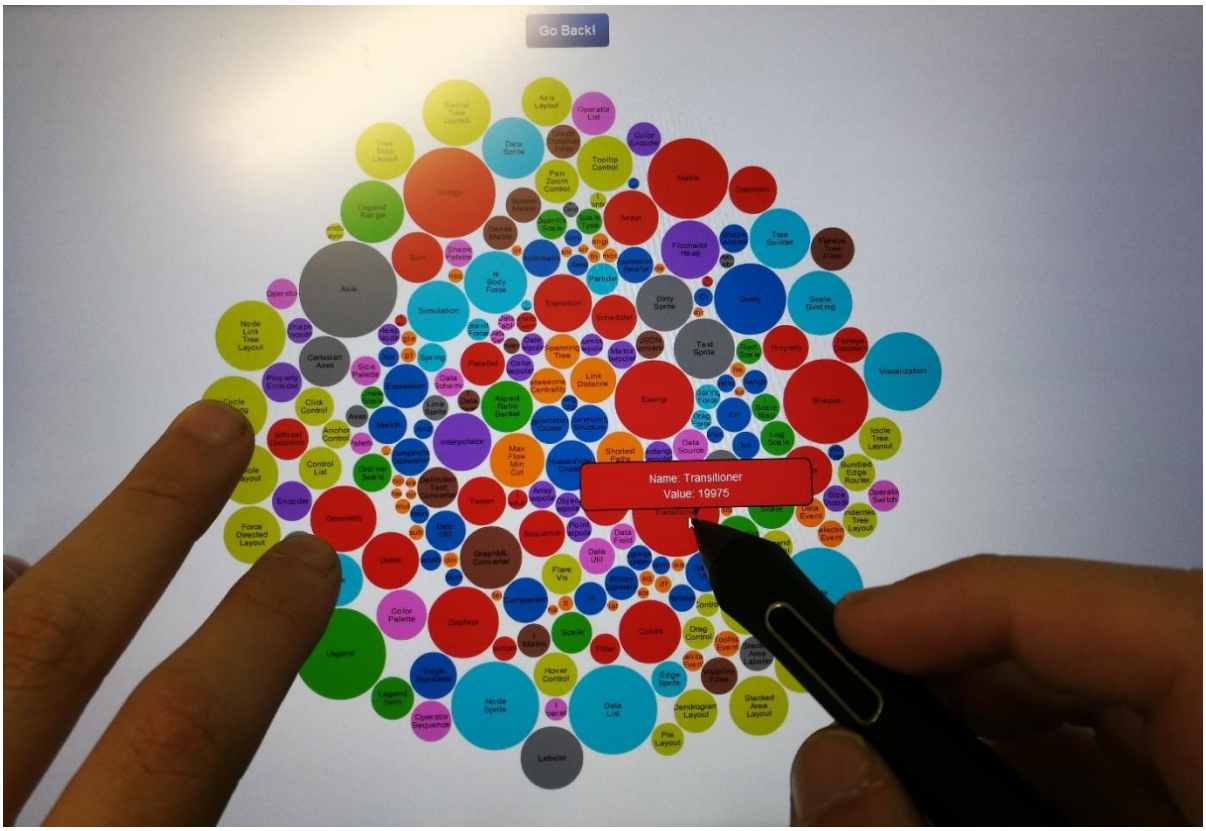


Figura 3.7 – Teste da interligação de leitura e de interpretação de dados com a representação de dados no ecrã do *tablet* gráfico.

Verificando-se o correto funcionamento da interligação dos dados com o sistema de representação de dados, passou-se ao desenvolvimento de uma interação mais complexa nas visualizações de dados já existentes e adicionar outras representações de dados com uma

componente de interação bimanual. Ainda por esta altura, grande parte dos testes das visualizações de dados ocorria dentro do navegador Mozilla Firefox em detrimento do navegador do Awesomium. Isto porque o Mozilla Firefox permite uma maior facilidade e rapidez de testes das visualizações.

O que este modo de testar não tem em conta é que as versões de JavaScript usadas pelo Firefox e pelo Awesomium não são as mesmas, sendo a versão do Awesomium a mais desatualizada. O que levou a que uma das visualizações que tinha sido desenvolvida usando a versão cinco da D3.js não pudesse ser executada no Awesomium. Devendo-se ao facto que algumas funções de JavaScript mais recentes não são suportadas pelo motor de renderização de páginas *web* do Awesomium. Sendo, assim, necessário fazer um retrocesso na versão da D3.js, descendo para a versão quatro deste. Esta mudança de versão obrigou a algumas alterações no modo como estaria escrita as chamadas a esta biblioteca de visualização de dados e foram removidas todas as chamadas de funções seta, visto que estas não são suportadas.

Neste momento a implementação do protótipo continha uma implementação básica da leitura de dados do *tablet*, faltando o acesso a mais alguns dados. Em termos de interação já era possível interagir com os dedos e com a caneta em simultâneo. No entanto a utilização dos dedos fazia com que o apontador do Windows saltasse entre os dedos e a caneta, isto porque a caneta e os dedos estavam a ser interpretados como apontadores do sistema. Os dados da caneta e dos dedos eram enviados para o Awesomium através da simulação de eventos de rato e teclado.

3.4.1 Recolha dos dados relativos às interações de toque e de caneta

Esta subsecção corresponde a uma descrição mais detalhada do módulo de acesso e armazenamento de dados do *tablet*, presente na Figura 3.1 da secção referente à arquitetura de sistema do protótipo desenvolvido.

Os dados a utilizar, relativamente às entradas de toque e de caneta, estão presentes em duas API separadas, sendo estas as API do bbTablet e a do Wacom Feel Multi-Touch que realizam, respetivamente, o acesso aos dados relativos à caneta e de cada dedo que esteja a pressionar o *tablet* gráfico.

Para aceder à API do bbTablet é necessária a criação de uma instância *singleton* do bbTablet na qual é preciso fazer a ligação a uma janela do gestor de janelas do Windows, para que esta possa recolher dados sobre a posição da caneta em relação a esta janela. Após a inicialização desta API é possível verificar se este conseguiu se conectar com sucesso ao *tablet* gráfico, uma vez que se não foi possível estabelecer a conexão então é necessário conectar novamente o *tablet* ao computador. Verificando-se que o *tablet* está conectado então pode-se proceder ao acesso aos dados desta API, este acesso é realizado através de duas formas distintas: por *callback* assíncronas[3] ou acedendo ao último evento reportado pelo *tablet*. Quando um utilizador aproxima a caneta do ecrã, se estiver a ser usado o sistema de *callbacks* do bbTablet, ocorre um atraso no reportar de informação, uma vez que a geração de novos eventos de *callback,* por vezes, tem uma velocidade superior à de tratamento das mesmas. Gerando uma fila de espera de eventos por tratar fazendo com que a informação disponibilizada não seja a mais atual, o que para um sistema deste género é muito prejudicial. No entanto se em vez de *callbacks* se for utilizada uma função que apenas acede ao último evento reportado pelo *tablet* gráfico, então este problema não ocorre porque não há mais informação a ser gerada do que a capacidade do computador a consumir. A desvantagem é que há uma ligeira perda de dados pois a leitura de eventos não é sequencial.

O acesso à Wacom Feel Multi-Touch API é efetuado depois de executar uma função, disponibilizada pela Wacom, que faz com que as sessões de acesso ao *tablet* por parte desta API sejam encerradas. Esta operação inicial é feita para que, mesmo que não exista outra sessão desta a correr, possa-se garantir que a inicialização desta é feita sem qualquer problema de concorrência aos recursos do *tablet*. Após este passo inicial, procede-se à inicialização da API em si, em que é feita a verificação da existência de *tablets* gráficos conectados e é feito um armazenamento de todos os dispositivos Wacom encontrados e as suas capacidades.

Desta API são utilizadas três funções de *callback*, sendo uma delas destinada para o caso de ocorrer a conexão de mais um *tablet* e outra para o caso de remoção, a terceira e última função de *callback* é uma função que irá ser chamada sempre que ocorrer uma deteção da

[3] Uma *callback* assíncrona é um tipo de *callback* não bloqueante, isto é, o sistema pode continuar a operar sem que a execução desta função (ou método) impeça as outras de serem executadas. Este tipo de *callbacks* é mais utilizado em sistemas que lidam com um sistema de entradas e saídas em que a chamada desta função é feita pelo sistema operativo.

presença de dedos a premir o ecrã do *tablet*, sendo, assim, esta a função a que irá realizar o tratamento dos dados reportados pelo dispositivo. Pelo facto de esta API funcionar através de *callbacks* deixa de existir a necessidade de estar constantemente a fazer pedidos de dados, sendo este processo totalmente automatizado, e só ocorrendo chamadas de execução das funções de *callback* se ocorrer alguma alteração relativamente ao que cada uma destas está incumbida de tratar, isto é, uma função de *callback* incumbida de tratar eventos de entrada de toque só será executada quando existir algum evento novo a ser tratado, não estando continuamente a correr. O sistema de *callbacks* desta API, ao contrário da anterior, não apresenta qualquer atraso nem uma quantidade excessiva de *threads* criadas para tratar todas as *callbacks* que sejam executadas.

Os processos de inicialização e utilização das API, descritos acima, estão implementados numa classe que para além de realizar estas configurações também é responsável por armazenar os dados recebidos da API bbTablet, convertendo alguns sem que haja perda de informação, numa estrutura de dados, existente apenas para guardar dados relativos à caneta, que é mais simples de aceder e que apenas contém os dados pretendidos, descartando os restantes para reduzir a quantidade de memória utilizada bem como a complexidade da estrutura de dados. Os dados recebidos através da Wacom Feel Multi-Touch API não sofrem qualquer conversão sendo apenas feita uma cópia dos dados reportados por esta para dentro de uma estrutura de dados, exclusiva para dados sobre os dedos, em que apenas estão presentes os dados que sejam necessários, pela mesma razão referida anteriormente. Estas duas estruturas de dados são, posteriormente, armazenadas dentro de uma terceira estrutura de dados que tem como propósito facilitar o envio destes dados, porque ao estar a informação toda presente numa só estrutura se torna mais simples a passagem de dados como retorno de funções, quando pedidos por código externo a esta classe.

Esta classe para além de armazenar os dados recebidos é também responsável por criar uma estrutura de dados temporária que é passada como retorno de uma função. Esta função existe com o objetivo de passar os dados de dentro desta classe para outras classes que irão proceder ao tratamento dos dados. Após esta função ser executada é imperativo correr outra função em que a estrutura de dados temporária é passada como argumento, para que esta seja eliminada da memória. Se a estrutura de dados não for eliminada e ocorrerem vários

pedidos sem que as outras estruturas de dados temporárias sejam eliminadas, encontra-se numa situação em que, devido à acumulação de dados desnecessários, poderá ocorrer uma exceção fatal devido a um vazamento da memória, sendo que este incremento de utilização de memória acontece a um ritmo de 1 Mb/s.

Na Figura 3.8 pode-se observar todos os dados que este módulo armazena e subsequentemente partilha com outros módulos.

```
Pen Data:
        Window X: 3485 Y: 597
        Raw X: 54570 Y: 13323 Z: 244 Pressure: 0
        System X: 3519 Y: 654
        Normalized Pressure: 0 Max: 0.49144
        Raw Pressure: 0
        Tilt X: 0.984809 Min: 0.145539 Max: 1
        Tilt Y: -0.173648 Min: 0.145539 Max: 1
        Angle: 0.140351 Angle in degrees: 8.42105 Min: 0.122807 Max: 1
        Button Mask: 0
        The Pen is in an normal state!
        The Pen is outside of the attached window!
Finger Data:
[ FINGER no.1 ]
        Finger X: 3513 Y: 366
        Confidence: 1
        Finger Width: 54.8945
[ FINGER no.2 ]
        Finger X: 3436 Y: 777
        Confidence: 1
        Finger Width: 82.9952
[ FINGER no.3 ]
        Finger X: 3411 Y: 572
        Confidence: 1
        Finger Width: 62.7366
```

Figura 3.8 – Dados disponibilizados pela classe de acesso às API.

Na Figura 3.8 observa-se que para cada instância de toque detetada na superfície do *tablet* é possível obter as coordenadas cartesianas de cada uma, bem como o nível de confiança de que essa entrada é um dedo a tocar no ecrã. Relativamente à caneta, os dados extraídos, permitem saber as coordenadas cartesianas da caneta, a pressão que se esteja a exercer na caneta, qual das pontas da caneta está virada para o ecrã, que botões da caneta estão a ser premidos e se a caneta está dentro de uma dada janela do sistema.

3.4.2 Interpretação dos dados de toque e de caneta

Esta subsecção corresponde a uma descrição mais detalhada do módulo de interpretação de dados, presente na Figura 3.1 da secção referente à arquitetura de sistema do protótipo desenvolvido.

Após a recolha de dados, descrita na subsecção acima, estes dados são passados da classe que faz a recolha e o armazenamento destes para a classe, criada pelo openFrameworks, que realiza o tratamento de dados. Alguns dos dados passados, nomeadamente os dados sobre os dedos que estão a ser detetados no ecrã, para esta classe não sofrem qualquer conversão. Uma vez que estes dados são lidos pelo sistema e consoante a quantidade de dedos a premir o ecrã estes ativam ou desativam certas funcionalidades, no fundo agindo como se fossem *modifiers*[4]. Os restantes dados, quando permitido pelos *modifiers* implementados, sofrem uma conversão de valores numéricos para eventos de teclado. Os eventos de teclado são emulados em código ao chamar as funções do openFrameworks, que normalmente são chamadas quando é detetado algum evento de carregar bem como levantar uma tecla. As funções utilizadas para fazer esta emulação de teclado são: "*ofKeyPress*" e "*ofKeyRelease*". Um exemplo desta conversão é a transformação da pressão da caneta em vários eventos de teclado distintos, uma vez que à medida que o utilizador vai variando a pressão exercida com a caneta, são emitidos diferentes eventos de pressão de uma tecla do teclado. Estes eventos são transmitidos para o navegador de *web Chromium* embebido numa janela de OpenGL através do *plugin* ofxAwesomiumPlus.

Um aspeto interessante dos dados reportados por parte do *tablet* é que estes não utilizam um referencial cartesiano com o eixo y invertido, isto é, o eixo y aponta para baixo em vez de apontar para cima (ver Figura 3.9).

[4] Os modificadores (*modifiers*) são eventos, despoletados, geralmente, pelo utilizador que alteram o funcionamento de uma dada ação, ativando um modo de operação alternativo. Por exemplo ao carregar e aguentar premida a tecla "shift" e premirmos simultaneamente uma tecla contendo uma letra, fazendo com que a letra a ser mostrada no ecrã seja maiúscula em vez de minúscula.

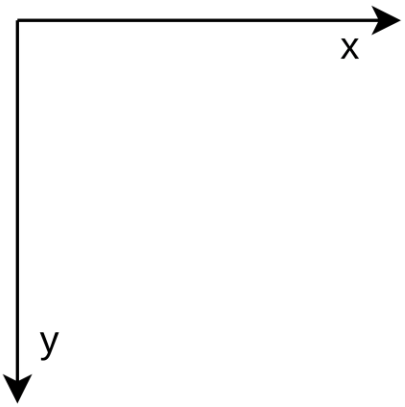


Figura 3.9 – Referencial cartesiano usado no *tablet*.

Este eixo invertido causa alguma confusão no mapeamento da inclinação da caneta pois ao inclinar a caneta para "cima" os valores decrescem, o que contraria as expetativas pois a mudança de valores ocorre no sentido inverso ao esperado. Devido a este eixo, a posição com as coordenadas (x, y) com os valores $(0, 0)$ está presente no canto superior esquerdo do *tablet.*

3.4.3 Renderização de visualizações de dados

Esta subsecção corresponde a uma descrição mais detalhada do módulo de renderização de visualização de dados presente na Figura 3.1 da secção referente à arquitetura de sistema do protótipo desenvolvido.

Para este trabalho, uma vez que este tem por base a representação de visualização de dados e posteriormente a manipulação destas, é necessário que se faça a passagem de dados para uma representação de dados visual, isto é feito através da utilização da biblioteca D3.js que quando utilizada dentro de um página HTML permite a representação gráfica dos dados quando este ficheiro é renderizado através de um renderizador de páginas HTML, podendo este ser um browser como o Firefox ou o Chrome.

No contexto deste projeto, a renderização de páginas *web* é realizada através da emulação de um *web browser* dentro do ambiente de renderização gráfica da OpenGL. O renderizador de páginas HTML a ser utilizado é o Chromium que se encontra dentro da biblioteca Awesomium que por sua vez é acedida pelo ofxAwesomiumPlus, que é um *plugin* para o openFrameworks.

Este *plugin* é capaz de carregar ficheiros HTML diretamente do computador através de um caminho absoluto para a localização do ficheiro no sistema ou através do acesso a um servidor de *web* através de HTTP. No entanto, apresenta a limitação de não conseguir executar funções de JavaScript se os ficheiros forem carregados diretamente do sistema, logo foi necessário a utilização de um servidor *web* para que fosse possível executar as funções de JavaScript presentes na página HTML.

Para o propósito de hospedar as páginas *web* localmente foi utilizado inicialmente o programa Served criado por (Johnson, 2016). Este é um simples programa que cria servidores à medida que pastas contendo ficheiros HTML vão sendo arrastadas para a janela deste programa, que por motivos de organização cada visualização de dados está contida numa pasta individual. O que levou à decisão de deixar de usar este programa e procurar outra solução foi o facto de cada pasta ter de ser carregada individualmente e para cada uma destas é criado um servidor numa nova porta o que fazia com que estas tivessem de ser carregadas por uma ordem específica, caso contrário, não coincidiria com os *links* utilizados dentro do openFrameworks.

Para solucionar este inconveniente e também de modo a que a preparação para inicializar o protótipo fosse mais simples e mais rápida foi utilizado a *framework* de criação de servidores Mongoose, que permite que se faça o carregamento de múltiplos ficheiros HTML de forma automática e num só servidor.

Como é possível observar na Figura 3.10, um servidor Mongoose é capaz de carregar múltiplas diretorias em simultâneo e, pelo facto de este gerar os *links* de acesso às páginas consoante o nome da diretoria em que estas se encontram, para aceder a estas é sempre através da mesma ligação.

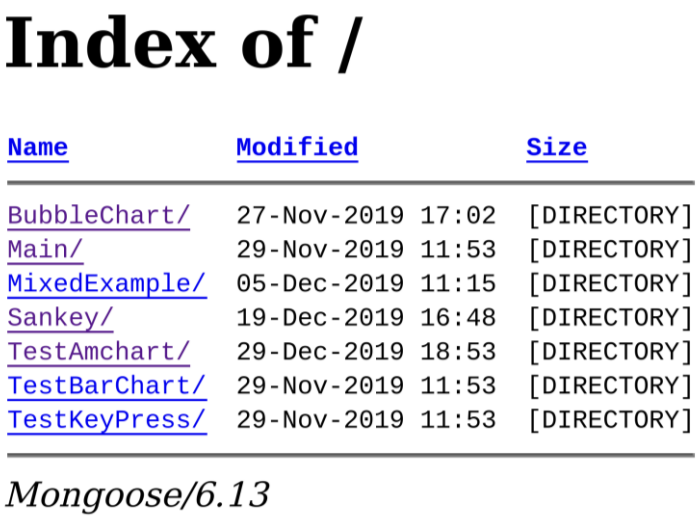


Figura 3.10 – Listagem de diretorias do servidor HTTP criado usando o Mongoose.

O facto de este servidor se encontrar na mesma solução de Visual Studio que o resto do protótipo permite que fique tudo mais centralizado. Também permite que quando for para testar o protótipo inicie-se os vários projetos dentro do Visual Studio em simultâneo, através da configuração de uma sessão de depuração. O que reduz o tempo de arranque do servidor para cerca de um ou dois segundos em vez de mais de um minuto quando o servidor Served estava a ser utilizado.

Com o servidor de HTTP a funcionar corretamente, o passo que se seguiu foi o de criar uma página HTML inicial só com funções de JavaScript básicas, sem a D3.js, visto que esta não vai apresentar nenhuma visualização de dados. Esta página serve como um menu de navegação e de listagem dos exemplos de visualização de dados disponíveis no servidor mencionado previamente.

O único senão que é encontrado aqui é que dentro do projeto de openFrameworks é necessário adicionar algum código para lidar com o carregamento de páginas, uma vez que não existe nada que indique qual a página que deve ser carregada após ter sido feito um clique num botão. Este botão que contêm uma referência para uma função dentro ofxAwesomiumPlus. Assim, ao clicar neste botão é despoletada uma mudança de estado que é interpretada pelo openFrameworks como um pedido de carregamento de página. Então, a mudança de estado desencadeia uma mudança de página no navegador de páginas *web* embebido no openFrameworks através do *plugin* ofxAwesomiumPlus.

Após este passo foi feito o download de exemplos de visualizações de dados com o formato da D3.js e inicialmente foi testado se os exemplos eram carregados de forma correta no browser do ofxAwesomiumPlus e pôde-se verificar que os exemplos eram executados e apresentados de forma correta dentro do openFrameworks, como se encontra exemplificado na Figura 3.11.

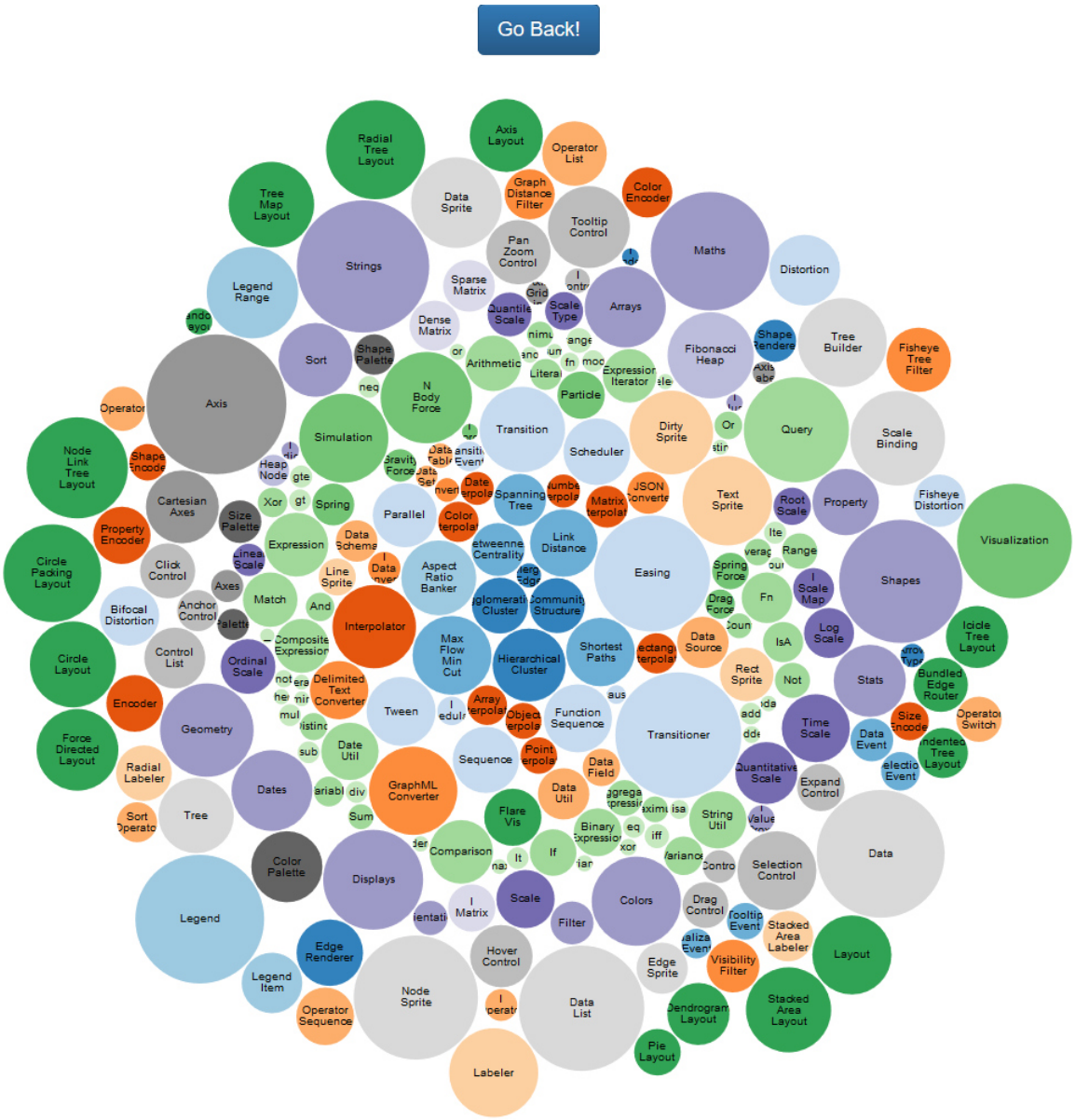


Figura 3.11 – Exemplo de uma página contendo uma representação de dados do tipo gráfico de bolhas.

Na Figura 3.11 é possível observar uma representação de dados do tipo gráfico em bolha e por cima desta está presente um botão que permite voltar atrás para o menu que permite a escolha das visualizações de dados a serem carregadas.

Com a confirmação do correto funcionamento da visualização das representações de dados, pôde dar-se como concluída a construção do processo de renderização de visualizações de dados. Este processo está ilustrado na Figura 3.12.

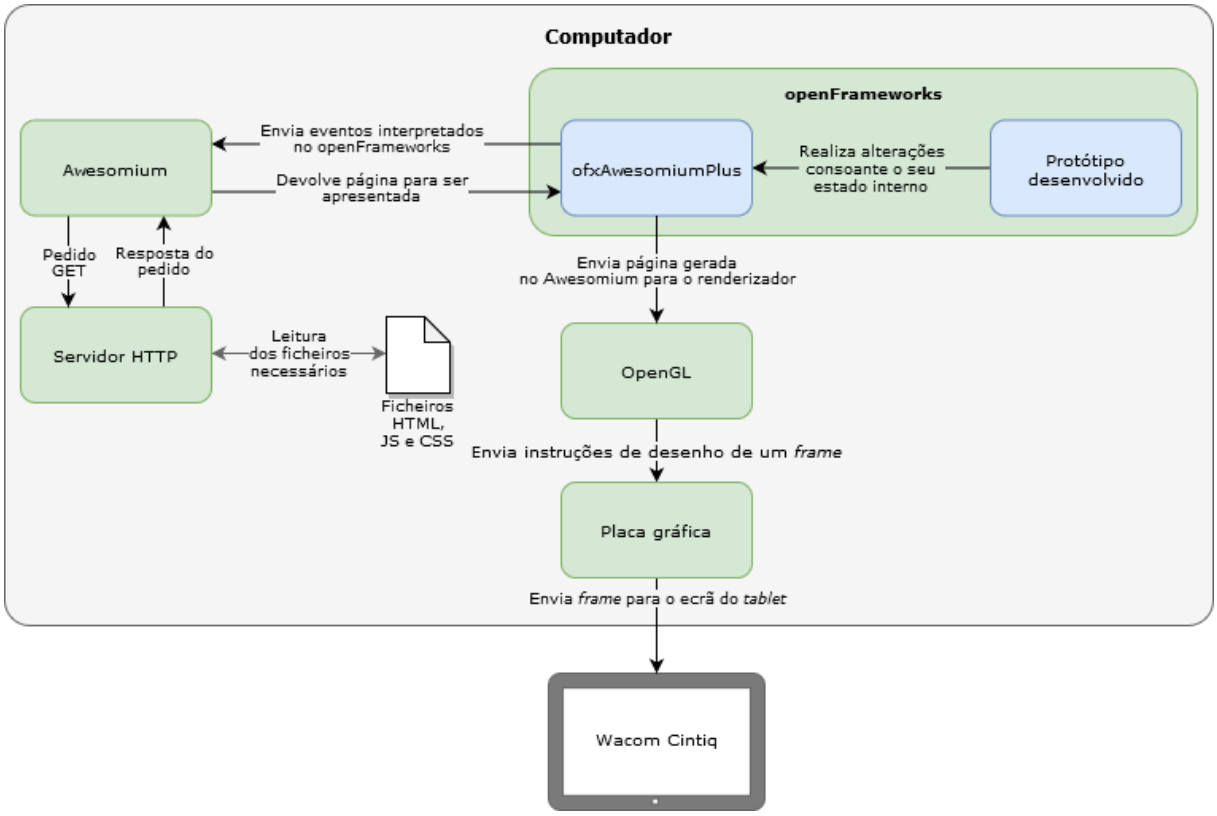


Figura 3.12 – Representação do processo de renderização.

Quando o servidor recebe um pedido GET para uma certa página contendo um *script* de D3.js, este obtém os ficheiros HTML contendo o *script* e os dados a serem representados e envia-os para o motor de renderização de páginas *web* Chromium, contido dentro do ofxAwesomiumPlus, que renderiza o ficheiro HTML numa página *web* que é vista como uma imagem estática. De modo a demonstrar as alterações ocorridas, devido à interação do utilizador, o openFrameworks renderiza várias imagens por segundo. Após este processo dentro do ofxAwesomiumPlus, uma das imagens contendo a página *web* é enviada para a OpenGL que instrui a placa gráfica em como renderizá-la no ecrã. A placa gráfica do computador, por sua vez, converte a informação recebida em pixéis, desenhando os *frames* que constituem tudo aquilo que é para ser mostrado no ecrã do *tablet*, quer seja a representação de dados quer seja outros elementos presentes no ambiente de trabalho, como por exemplo o ponteiro indicador do sistema operativo. Após ter sido feita a renderização do

frame, este é enviado para o *tablet* Wacom, através de um cabo HDMI, onde finalmente o utilizador pode visualizar a representação de dados pretendida.

3.5 Métodos de interação

Neste subcapítulo será explicado e demonstrado como é que o utilizador interage com o protótipo e como este pode interagir com cada tipo de visualização de dados implementada.

O protótipo desenvolvido para este trabalho tem como método de interação a interação bimanual de toque e de caneta sobre um *tablet* gráfico Wacom. Para poder interagir com as representações de dados apresentadas foram criadas algumas modalidades de interação, sendo elas a pressão exercida na ponta da caneta, a inclinação desta em relação ao ecrã do *tablet*, os dedos a pressionar no ecrã e o clique da caneta no ecrã. Os métodos de interagir com as representações através da pressão e da inclinação da caneta são passados para o navegador *web* como eventos de teclado, isto é, é simulado o pressionar de uma tecla do teclado. O clique da caneta no ecrã é convertido para um evento de clique do botão principal do rato, nesta conversão as coordenadas do *tablet* em relação ao sistema são convertidas para as coordenadas internas da aplicação, nesta conversão é assumido que o *tablet* possui uma resolução FHD e que está à direita do ecrã principal do computador. Os dedos a premir no ecrã agem como *modifiers*, ativando e desativando os métodos de interação anteriores conforme for necessário. Por exemplo, quando é ativada pressão no ecrã é desativada a emulação de cliques no ecrã.

3.5.1 Interações multimodais desenvolvidas

Nesta subsecção serão expostas as interações bimanuais de toque e de caneta e uma interação monomanual com a caneta desenvolvidas como método de interação sobre o sistema. Cada uma destas interações despoleta uma ação na visualização, no entanto estas ações executadas dependem da visualização a ser apresentada, por outras palavras, para visualizações diferentes a mesma interação tem diferentes ações associadas.

A interação mais simples de realizar é uma ação análoga à ação de clicar com o botão primário do rato. Esta ação, representada na Figura 3.13, consiste em tocar com a ponta da

caneta no ecrã. Esta é uma interação monomanual, visto que apenas é necessário usar uma mão, geralmente a dominante, para realizar esta ação.

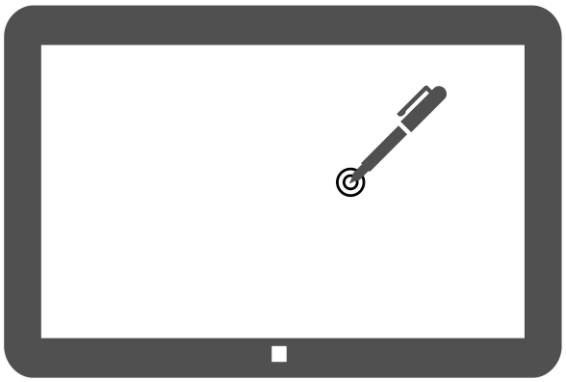
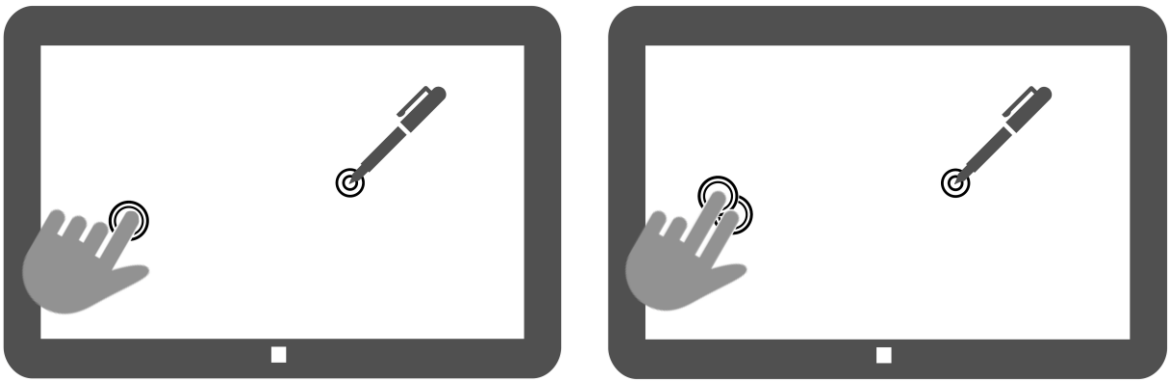
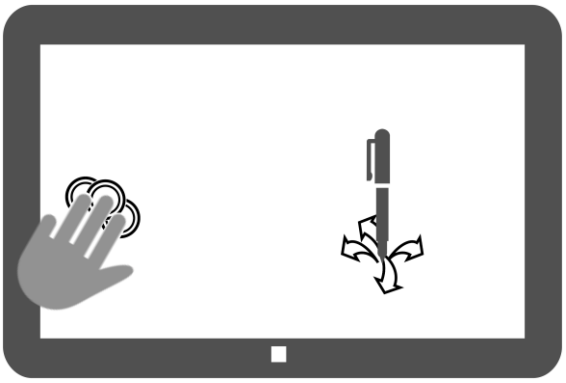


Figura 3.13 – Interação monomanual de seleção.

As interações bimanuais são as seguintes: um dedo a premir sobre o ecrã e premir com a ponta da caneta sobre o ecrã fazendo variar a pressão sobre a mesma, como se pode ver na Figura 3.14 a), a segunda interação criada foi de colocar dois dedos sobre o ecrã e novamente variar a pressão exercida na ponta da caneta (ver Figura 3.14 b)) e, por fim, três dedos a premir o ecrã e inclinar a caneta para a frente, para trás, para a esquerda ou para a direita, estando esta última interação representada na Figura 3.14 c).



a) b)



c)

Figura 3.14 – Interações bimanuais desenvolvidas: a) interação com o sistema usando um dedo e variando a pressão da caneta; b) interação usando dois dedos e variando a pressão; c) interação que envolve três dedos a premir o ecrã e a inclinação da caneta num dos eixos xy.

Estas interações que foram programadas dentro do projeto de Visual Studio desenvolvido, no qual, estas são interpretadas e convertidas em eventos de rato ou de teclado que, posteriormente, são transmitidos para o navegador Awesomium. Neste, os eventos são interpretados consoante o código de JavaScript e as funções de D3.js implementadas e são feitas alterações nas visualizações de dados. Nas seguintes subsecções serão apresentadas as visualizações de dados utilizadas e como é que um utilizador pode interagir com as mesmas.

3.5.2 Gráfico de bolhas

O primeiro tipo de gráfico a ser adicionado interação foi o gráfico de bolhas, representado na Figura 3.15. Neste gráfico, cada círculo representa uma extremidade de um diagrama hierárquico, como por exemplo o da Figura 2.5. O tamanho de cada círculo corresponde à quantidade de linhas de código da classe que lhe está atribuída. Estes círculos encontram-se agrupados por cores, das camadas mais internas para as mais externas, consoante a classe mãe de cada círculo. Pode-se observar este agrupamento, na Figura 3.15, em que no centro do gráfico estão elementos com a mesma cor pois estes possuem todos a mesma classe mãe, e também pode-se observar que existem outros círculos, que não possuem a mesma classe mãe, com esta cor nas camadas mais externas, isto acontece devido à limitação do número de cores utilizadas. No centro de cada um destes círculos está presente o nome da classe a que este corresponde.

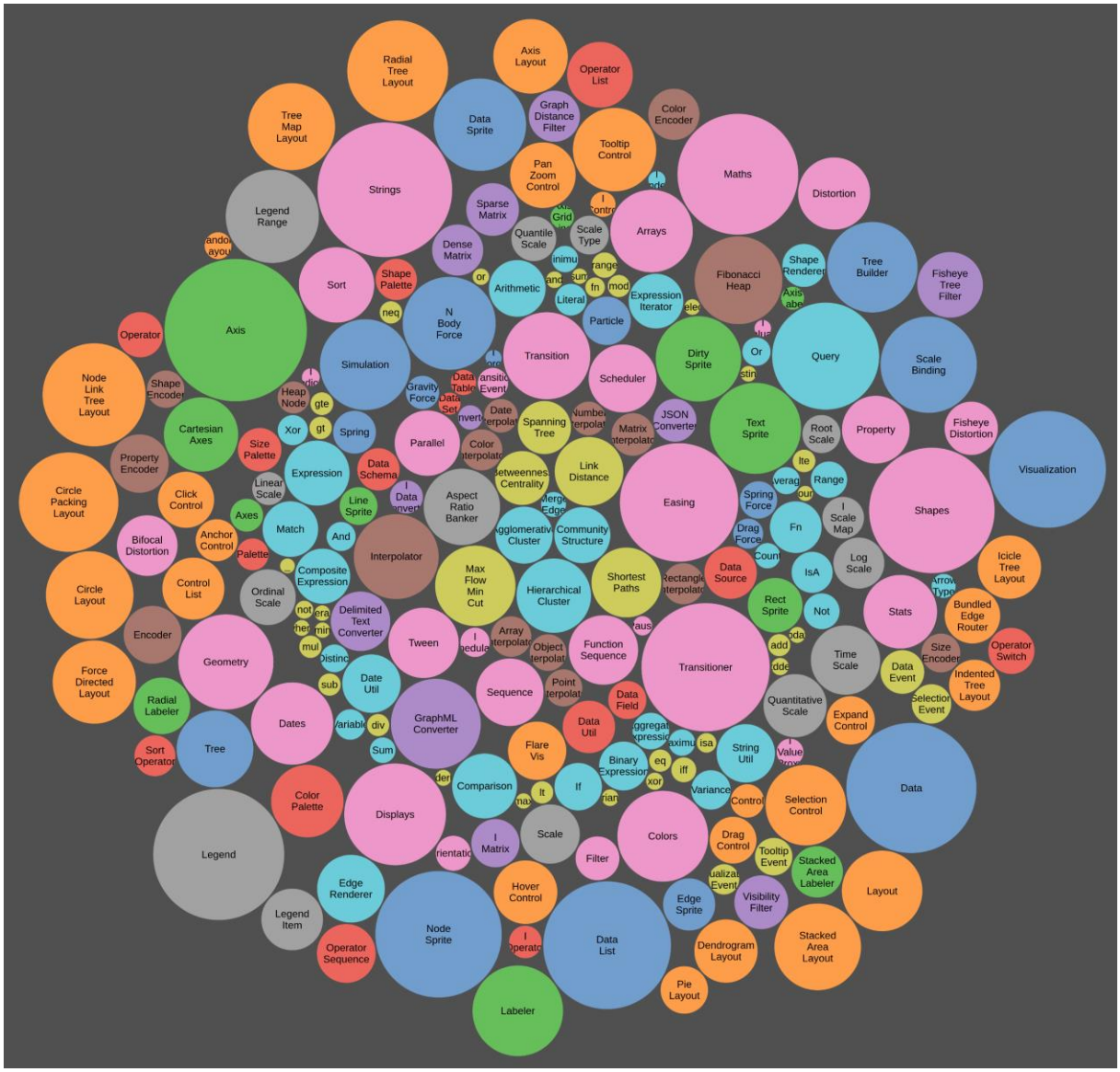


Figura 3.15 – Gráfico de bolhas representando a hierarquia de classes da D3.js.

Este gráfico apenas possui três métodos de interação sobre este. Para este tipo de gráfico foram desenvolvidas três ações de interação. Destas três ações apenas duas são do tipo bimanual, dado que uma é apenas o clique com a ponta da caneta sobre uma bolha do gráfico representado no ecrã. O utilizador ao realizar esta ação, faz com que apareça por cima do círculo premido um *pop-up* contendo o nome daquela entrada de dados e o valor atribuído a esse, na Figura 3.16 encontra-se um exemplo deste *pop-up*. Esta ação também seleciona o círculo atual para que se possa aplicar as outras ações sobre este. O círculo selecionado apresenta um anel branco à sua volta.

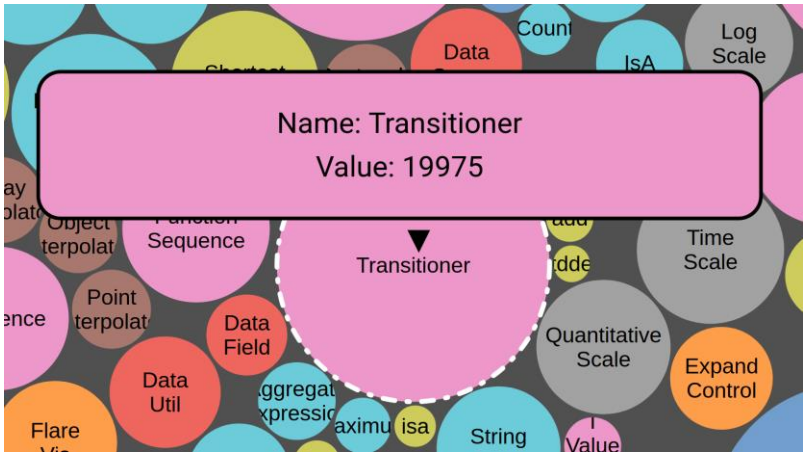
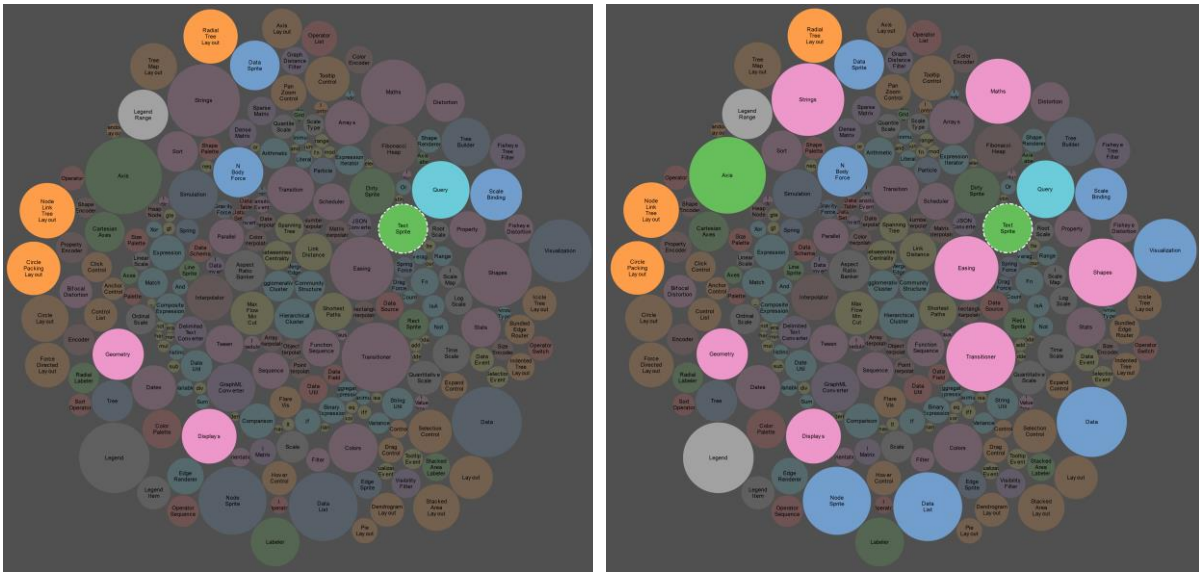


Figura 3.16 – *Pop-up* que aparece quando o utilizador realiza a ação de clicar sobre o círculo "Transitioner".

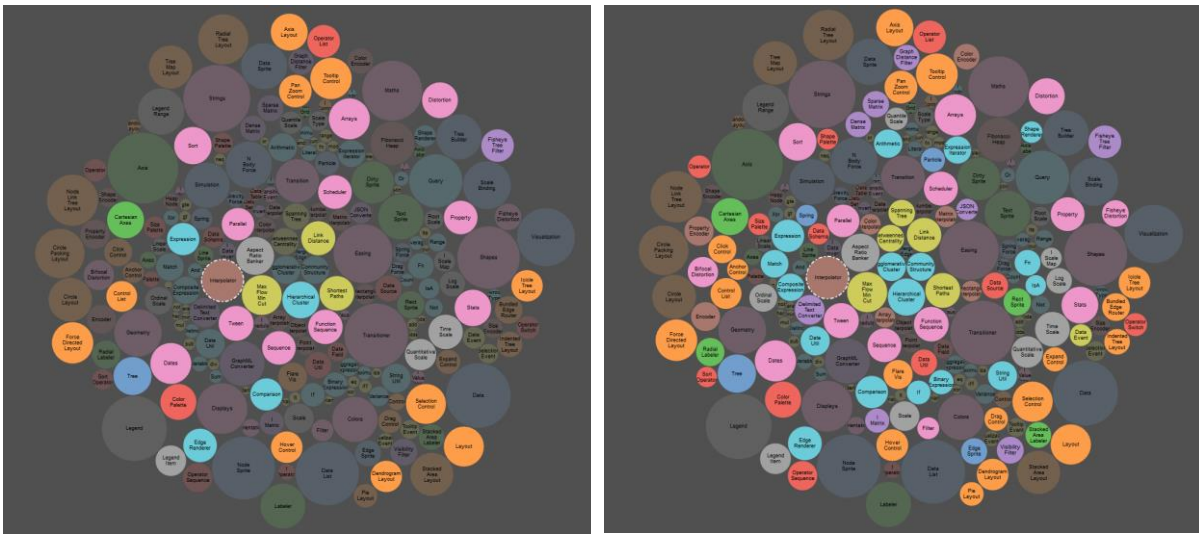
As outras duas ações disponibilizadas para este tipo de gráfico são ambas do tipo bimanual, sendo que para realizar esta ação o utilizador tem que, obrigatoriamente, realizar a ação de clicar sobre o círculo. Em ambas as ações bimanuais o utilizador tem de fazer variar a pressão da caneta ao mesmo tempo que prime no ecrã com um ou dois dedos. O utilizador, ao premir com apenas um dedo, ativa o modo em que, a pressão exercida na ponta da caneta, faz com que sejam escondidos alguns elementos que tenham um valor que seja inferior ao valor do círculo selecionado bem como alguns elementos que tenham um valor superior e que não façam parte do intervalo de valores estabelecido pela pressão exercida. Este modo é, então, um modo de pesquisa por círculos com valores superiores ao selecionado, este modo de pesquisa está representado nas Figura 3.17 a) e b). Na Figura 3.17 a) tem-se a pesquisa por valores superiores com o utilizador a exercer pouca pressão na ponta da caneta e na Figura 3.17 b) apresenta-se o resultado de fazer variar a pressão para mostrar mais círculos com um valor superior.



a) b)

Figura 3.17 – Destaque das classes com uma quantidade de linhas superior à selecionada. Quantidade de pressão a ser exercida na ponta da caneta: a) menos pressão; b) mais pressão.

Se o utilizador realizar uma ação semelhante à descrita anteriormente, mas premindo no ecrã com dois dedos em vez de apenas um, então é ativado o modo de pesquisa por círculos com um valor inferior ao círculo selecionado. As Figura 3.18 a) e b) mostram o mesmo processo que o descrito para uma pesquisa por valores superiores, mas aplicado para valores inferiores aos do círculo selecionado.



a) b)

Figura 3.18 – Destaque das classes com uma quantidade de linhas inferior à selecionada. Quantidade de pressão a ser exercida na ponta da caneta: a) menos pressão; b) mais pressão.

Nestas ações a pressão da caneta faz variar uma percentagem p num intervalo de valores $[30, 100]$, subindo apenas de 10 em 10 valores. Esta variação de valor faz com que o resultado da equação (3.1) varie entre $[V_c, r]$ se se estiver à procura de círculos com valores superiores

ou $[r, V_c]$ à procura de valores inferiores ao do círculo selecionado. Nesta equação V_c é o valor do círculo escolhido, p a percentagem indicada pela pressão exercida na ponta da caneta, V_d o menor ou maior valor presente nos dados consoante o utilizador coloque um ou dois dedos no ecrã do *tablet* e r o resultado da operação da equação (3.1).

$$r = V_c + (p * (V_d - V_c)) \tag{3.1}$$

3.5.3 Diagrama de Sankey

Este tipo de gráfico, representado na Figura 3.19, apresenta vários nós que têm algumas ligações entre si, cada nó representa uma fonte se este não tiver ligações à entrada, uma utilização ou transformação se tiver ligações de entrada e de saída ou um destino se não possuir ligações de saída. Neste caso, cada nó representa um método de criação de energia elétrica, transformação ou o seu destino. Em cada nó pode-se ter uma ou mais ligações de entrada e/ou saída que estejam ligadas a outros nós, sendo que a soma de todos os valores à entrada tem de corresponder ao valor que é atribuído à saída e vice-versa, excetuando os casos em que um nó não possua ligações de entrada ou de saída. Estas ligações possuem uma largura que corresponde ao valor que é passado de um nó para outro. A largura de um certo nó corresponde sempre à largura de todas as suas ligações à entrada ou à saída. Neste diagrama, as cores não possuem qualquer significado, existindo apenas para tornar mais fácil a visualização de dados distintos.

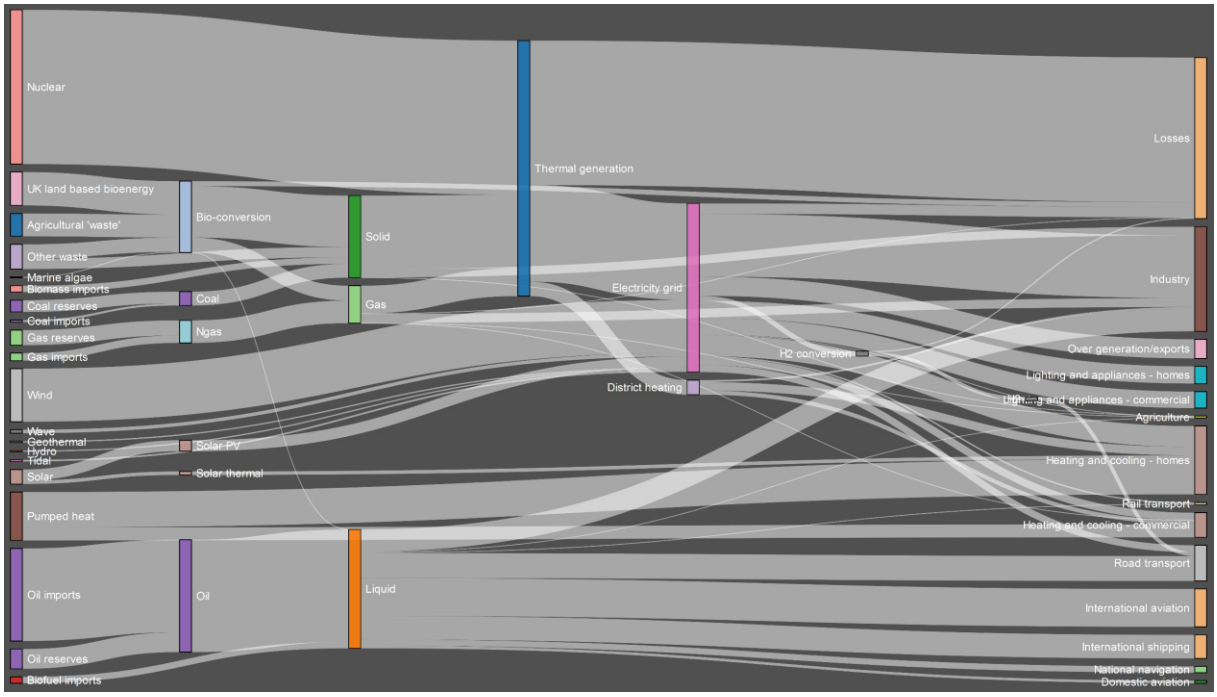
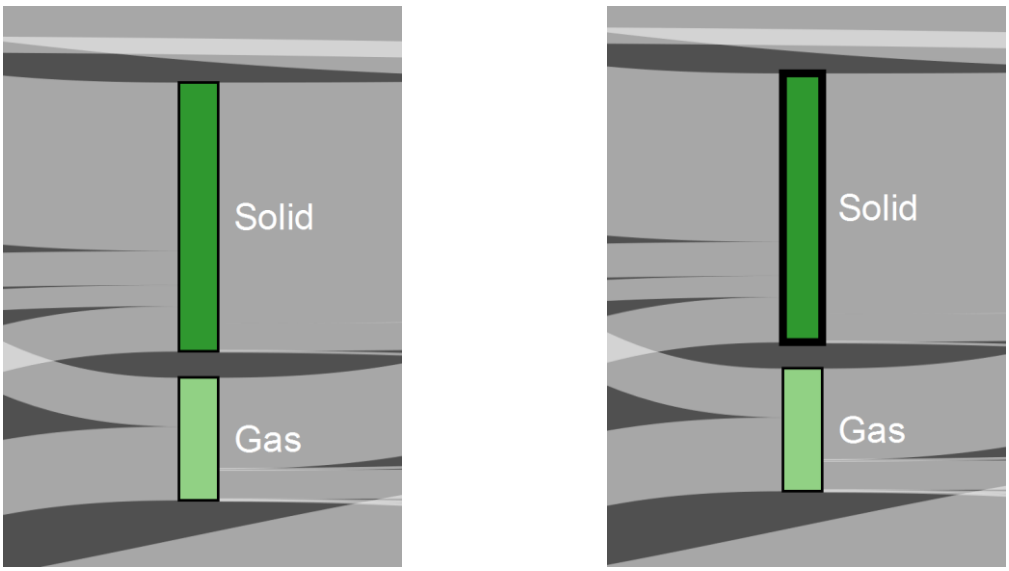


Figura 3.19 – Diagrama de Sankey.

Para este tipo de visualização foram desenvolvidos quatro métodos de interação. O primeiro método é o de seleção de nó em que o utilizador ao premir com a ponta da caneta sobre um dado nó, por exemplo o nó "Solid" presente na Figura 3.20 a), faz com que este seja selecionado. Esta seleção é representada por uma linha preta em volta do nó selecionado, representado na Figura 3.20 b). Esta ação é a única sobre esta representação de dados em que não é utilizada uma interação bimanual.



a) b)

Figura 3.20 – Seleção de nós: a) dois nós não selecionados; b) nó "Solid" selecionado.

O utilizador ao premir com um dedo e ao pressionar com a ponta da caneta em simultâneo no ecrã faz com que, se houver algum nó selecionado, seja mudado as cores das ligações de saída para a cor do nó, como demonstrado na Figura 3.21. Se não houver um nó que esteja selecionado, então não acontece nada.

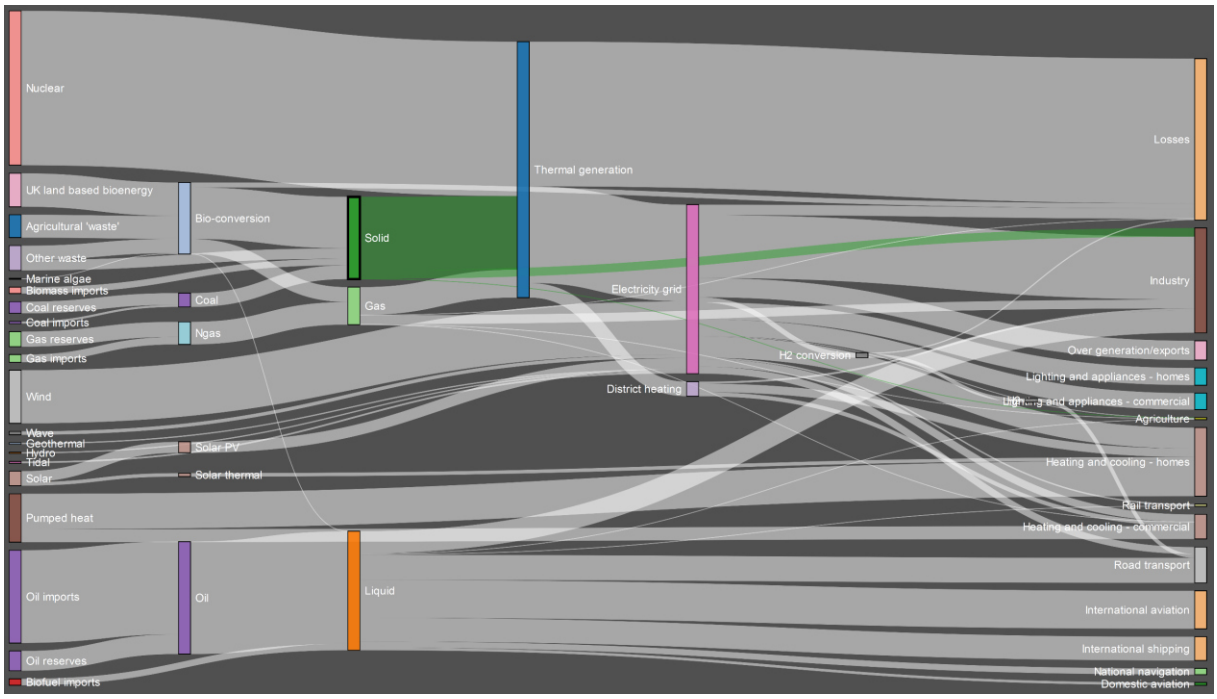


Figura 3.21 – Ligações de saída do nó “Solid” destacadas.

Se o utilizador premir com dois dedos e pressionar com a ponta da caneta em simultâneo no ecrã é realizada uma ação semelhante à descrita no parágrafo anterior. Sendo que a diferença é que esta ação muda a cor das ligações de entrada de um nó, representado na Figura 3.22. Em ambas as ações se as ligações a serem alteradas não existirem, então estas ações não modificam nada.

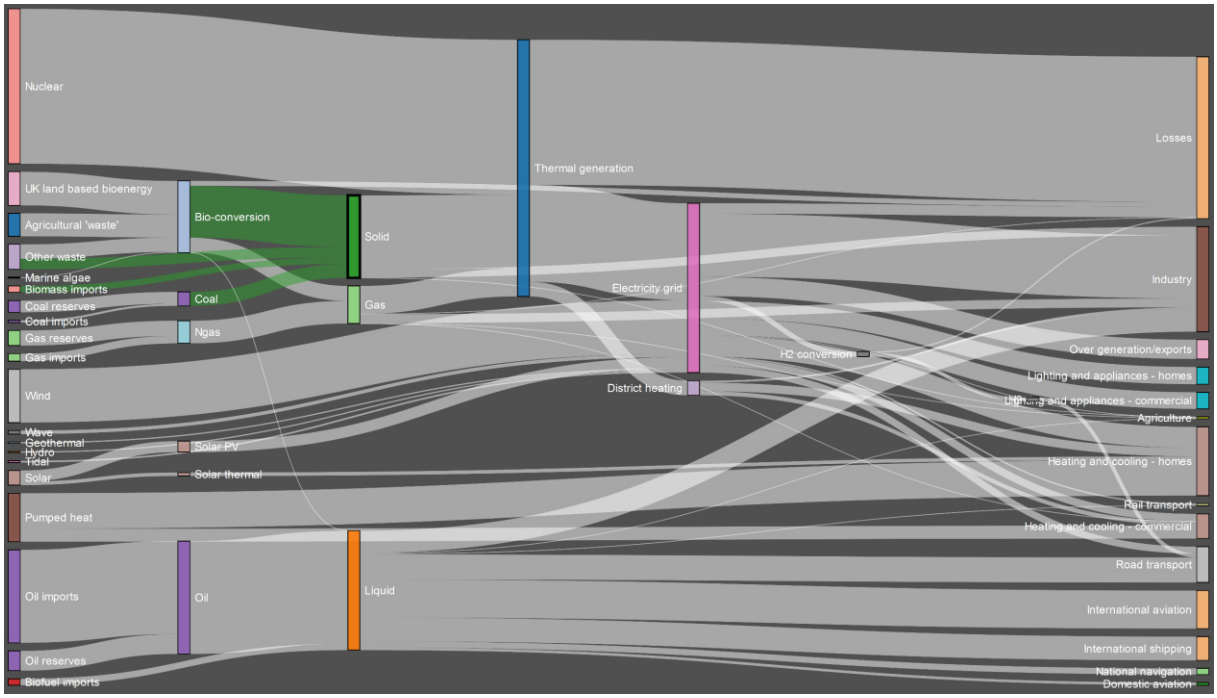
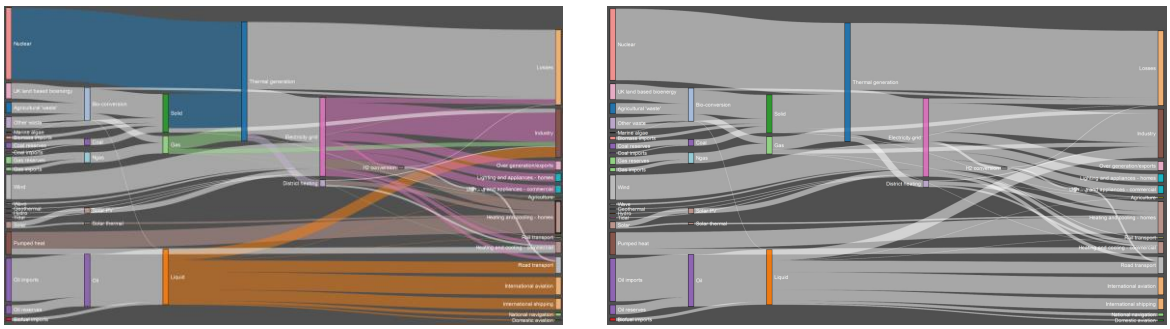


Figura 3.22 – Ligações de entrada do nó “Solid” destacadas.

A última ação desenvolvida para esta visualização de dados é uma ação que envolve o utilizador premir no ecrã com três e, com a caneta próxima do ecrã, inclinar a caneta para qualquer lado, isto é, inclinar a caneta para a frente, para trás, para a esquerda ou para a direita, faz com que as ligações que a cor tivesse sido mudada sejam repostas para a sua cor inicial e que algum nó que esteja selecionado deixe de estar selecionado (cf. Figura 3.23).



a) b)

Figura 3.23 – Reposição, para o estado original, do diagrama de Sankey: a) com várias ligações destacadas; b) após ter sido feita a ação para este ficar sem qualquer item destacado.

3.5.4 Gráfico de dispersão e de radar

No terceiro tipo de visualização de dados combinou-se dois tipos de gráficos, um gráfico de dispersão e um de radar. Como se pode ver na Figura 3.24, o gráfico de dispersão encontra-se do lado esquerdo do ecrã do *tablet* e o do tipo radar, sem qualquer elemento de dados representado neste, encontra-se no lado direito.

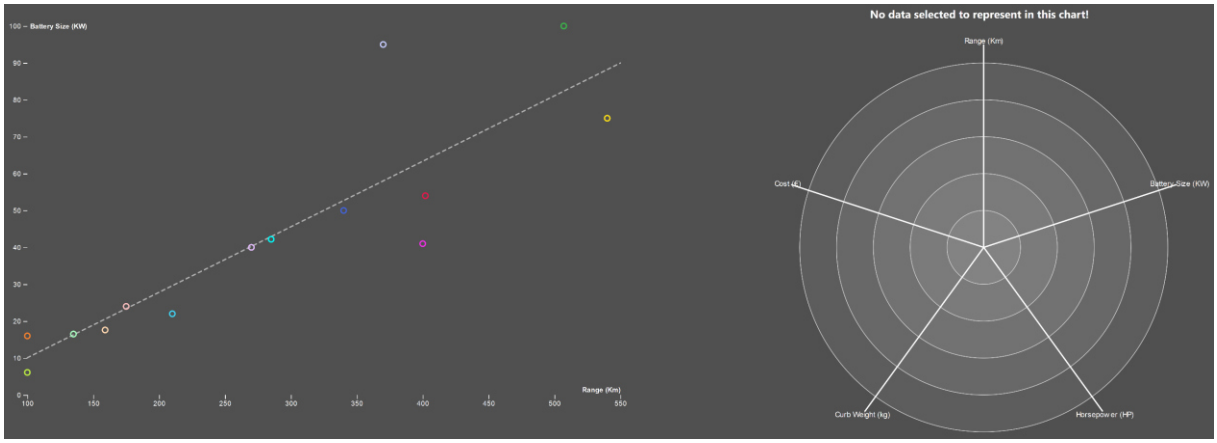


Figura 3.24 – Os gráficos de dispersão e de radar como são apresentados ao utilizador.

Na Figura 3.25 pode-se observar uma vista aproximada do gráfico de dispersão. Neste tipo de gráfico, os dados são representados num plano de coordenadas cartesianas consoante os seus valores. No caso representado na figura, o eixo x representa os valores do alcance de um veículo em quilómetros e o eixo y é a capacidade da bateria do veículo elétrico em quilowatt-hora. Cada par de valores xy é representado por um círculo. A cor de cada um destes serve para que seja mais fácil de distinguir cada um dos veículos representados. Também está presente uma linha descontínua que representa a regressão linear dos vários pares xy.

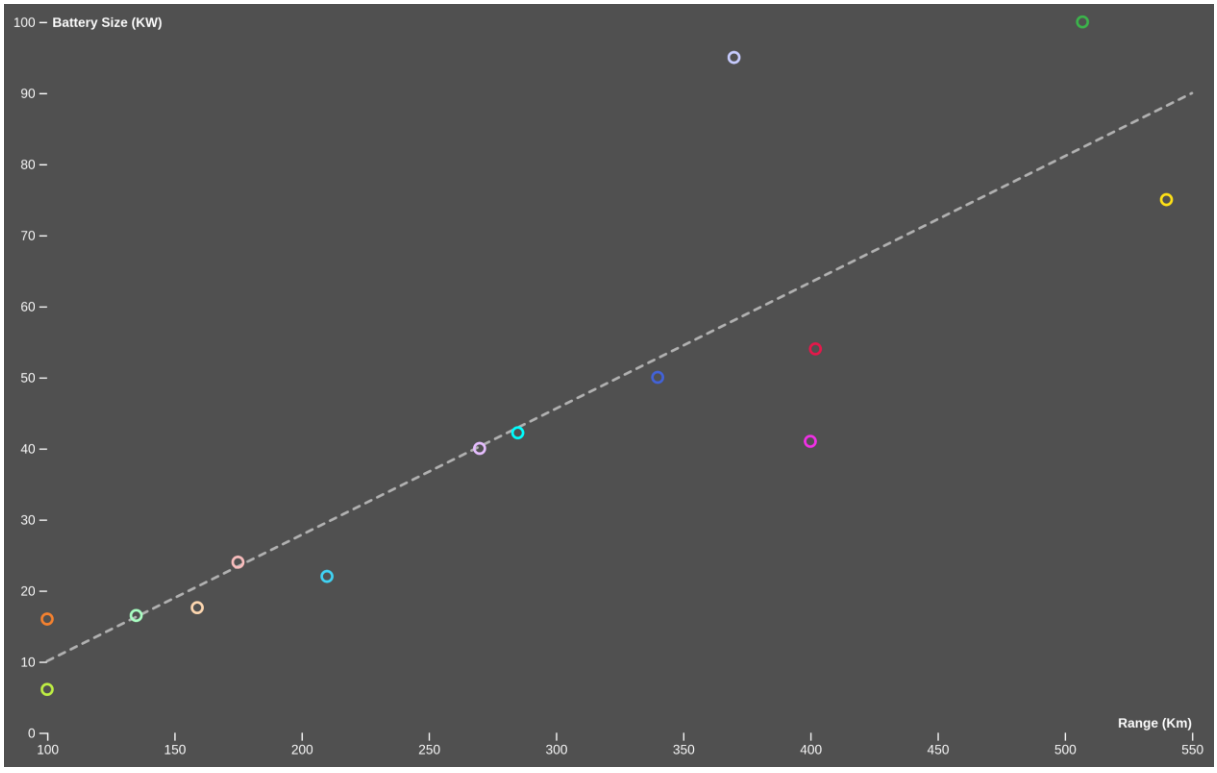
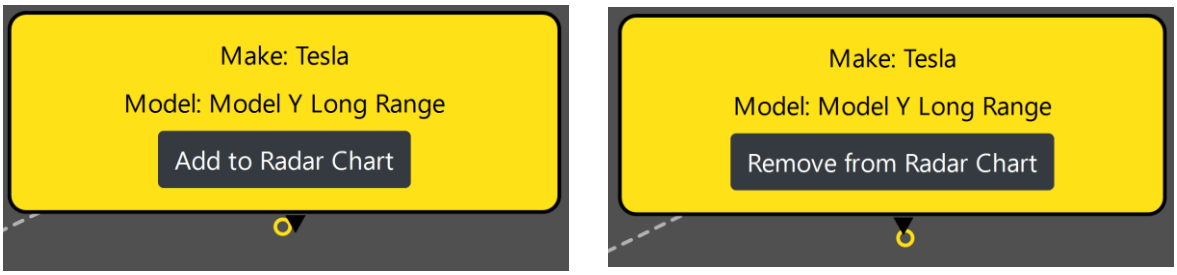


Figura 3.25 – Vista aproximada do gráfico de dispersão.

Sobre o gráfico de dispersão são disponibilizadas duas ações. A mais simples das duas é a de seleção em que o utilizador clica sobre um círculo e aparece um *pop-up* em que lhe é apresentado qual é o veículo selecionado e também um botão onde este pode escolher se adiciona ou remove os dados destes veículo do gráfico de radar, como podemos observar na Figura 3.26 a) e b), respetivamente.



a) b)

Figura 3.26 – *Pop-up* que aparece após utilizador pressionar sobre um círculo no gráfico de dispersão: a) opção para adicionar ao gráfico de radar; b) opção para remover do gráfico de radar.

A outra ação que é permitida sobre este gráfico é a que permite ao utilizador alterar que informação é representada nos eixos xy. A ação de colocar três dedos no ecrã e inclinar a caneta, com esta próxima do ecrã, no eixo x ou y. Ao inclinar para a frente ou para trás a variável no eixo y é modificada e ao inclinar para a esquerda ou para a direita o eixo x é modificado. O utilizador ao realizar esta ação, inclinando a caneta para cima ou para baixo, faz com que mude a variável apresentada no eixo y, por exemplo, partindo da Figura 3.25 e executando esta ação o gráfico atualiza e mostra uma nova variável no eixo y, como se pode ver na Figura 3.27.

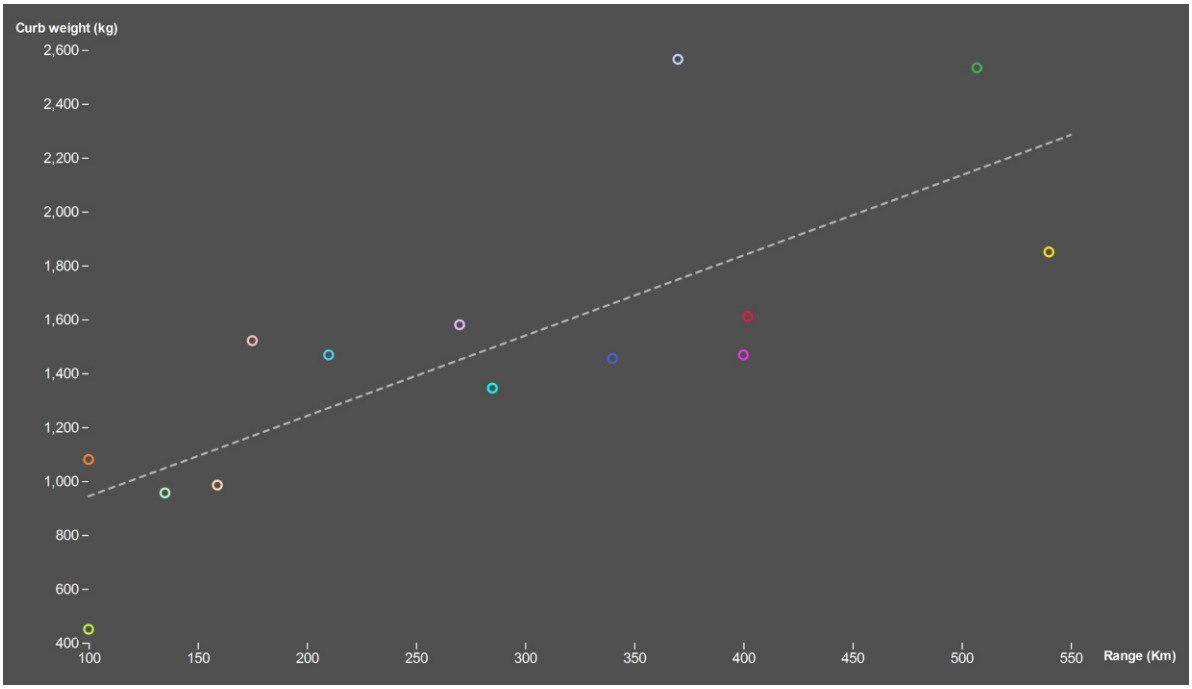
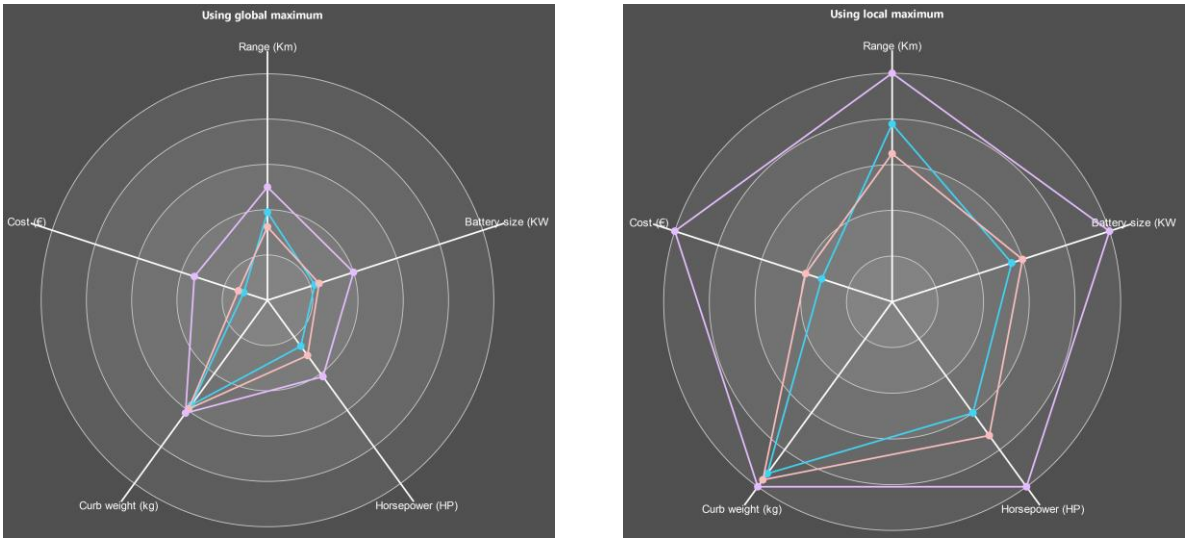


Figura 3.27 – Eixo y modificado pela ação de inclinar a caneta para cima ou para baixo.

Nos gráficos de radar, demonstrados nas Figura 3.28 a) e b), tem-se três veículos, cada um representado com uma cor distinta.



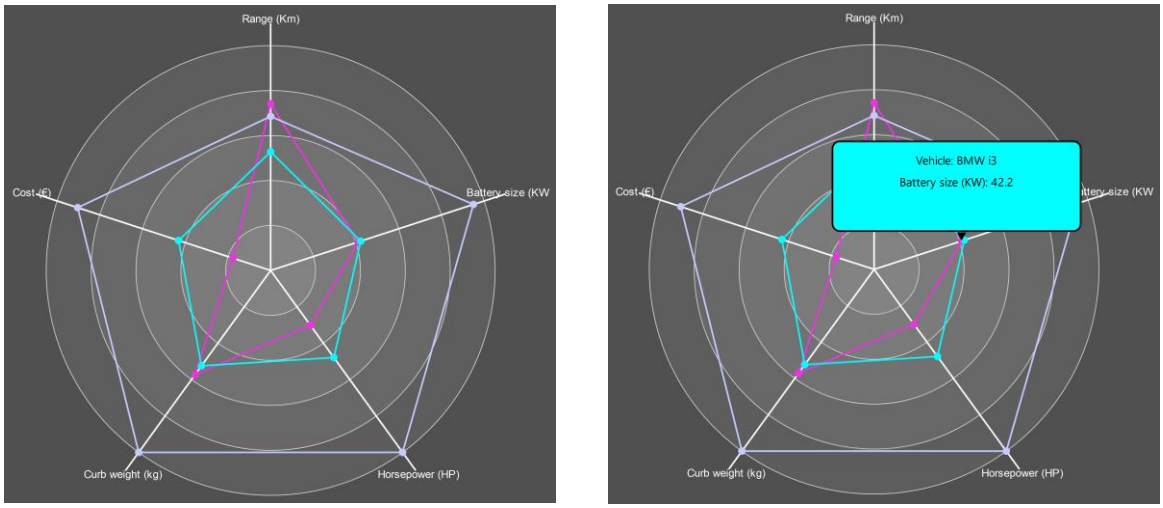
a) b)

Figura 3.28 – Vista aproximada do gráfico de radar, com os dados de três veículos representados. Usando uma escala de valores: a) máximos globais; b) máximos locais.

Estes três veículos foram adicionados através do *pop-up* representado na Figura 3.26 a). É possível observar que as representações de dados presentes nos gráficos são diferentes um do outro. Esta diferença deve-se a que no gráfico representado na Figura 3.28 a) todas as escalas utilizam valores máximos globais, isto é, são dimensionadas consoante os valores

máximos, de entre todas as entradas de dados. Na Figura 3.28 b) os valores máximos das escalas apenas têm em conta os valores máximos dos dados representados nestas.

O gráfico de radar possui três ações que podem ser efetuadas. A primeira ação envolve o utilizador passar por cima de um dos pontos de uma dada linha para revelar um *pop-up* que contém o nome do veículo e o valor atribuído a esse ponto, como se pode verificar na Figura 3.29. Podemos observar, na Figura 3.29 a), o estado inicial, para esta ação, do gráfico de radar e após o utilizador passar pelo nó da linha azul clarinha no eixo que indica o "Battery size (KW)", revelando mais informação sobre esse nó (ver Figura 3.29 b)).



a) b)

Figura 3.29 – Demonstração do *pop-up*: a) estado inicial do gráfico de radar; b) *pop-up* revelado através da ação de passar por cima do nó.

A segunda ação desenvolvida para este tipo de gráfico é aquela em que o utilizador prime com um dedo e com a ponta da caneta no ecrã em simultâneo muda o estado deste radar para o estado em que este apresenta os valores segundo uma escala de valores globais, como se confere na Figura 3.28 a).

Fazendo uma ação similar à anterior, com a diferença que o utilizador, nesta ação, prime com dois dedos. Esta interação faz com que o gráfico mude de estado para apresentar as escalas consoante valores máximos locais, representado na Figura 3.28 b).

Estas três visualizações de dados estão disponíveis na aplicação desenvolvida para este projeto e podem ser acedidas através de um menu inicial. Existindo a possibilidade de o utilizador navegar livremente entre as visualizações de dados criadas.

Após a criação das interações para cada visualização de dados, deu-se por concluído o desenvolvimento do protótipo, uma vez que já se tinha desenvolvido uma aplicação que era capaz de suportar a interação bimanual de toque e de caneta sobre visualizações de dados. Chegando-se ao fim desta fase, passou-se à avaliação do sistema desenvolvido.

4 Avaliação

Neste capítulo serão apresentadas a metodologia da avaliação, os resultados desta avaliação e uma breve discussão do que estes resultados implicam no contexto deste trabalho.

4.1 Metodologia

Para poder avaliar o funcionamento do sistema desenvolvido, foram escolhidas duas das visualizações para serem alvo de um processo de testes com utilizadores. As duas visualizações escolhidas foram o gráfico de bolhas e o gráfico de dispersão conjuntamente com o diagrama de radar, doravante referidas como visualização A e B, respetivamente.

Escolhidas as visualizações a serem usadas para testar o sistema, delineou-se como seriam feitos os testes com utilizadores. Para que se possa avaliar a qualidade deste sistema para interpretar os dados das visualizações, foi feito uma análise comparativa com um sistema similar em que a interação seja feita através do rato e do teclado. Sendo assim, foi pensado que cada participante da avaliação iria utilizar cada visualização duas vezes seguidas, uma vez usando o *tablet* através da caneta e dos dedos e outra na qual usaria o computador através do rato e do teclado.

Foi estipulada uma sequência de testes para os utilizadores, em que cada um tenha as condições de teste diferentes dos demais, de forma a evitar que os utilizadores, ao longo do processo de avaliação, demonstrem um efeito de aprendizagem ou de novidade. Estes efeitos podiam fazer com que um utilizador se sinta mais à vontade com um sistema do que o outro, fazendo com que os resultados dos testes não sejam fidedignos. As condições de teste são indicadas por letras:

- em relação às visualizações, a letra A indica que a primeira visualização a ser testada pelo utilizador será a visualização A e a letra B será a visualização B;
- para o método de interação, definiu-se que a letra R indica que, para a uma dada visualização, a primeira modalidade de interação testada foi a interação com o rato

e o teclado e só depois o *tablet*. A letra T indica que a primeira modalidade testada foi a interação com o *tablet* e, posteriormente, a interação com o rato e o teclado.

Um exemplo de uma combinação foi a BTT. Esta combinação indica que o participante começou a testar o sistema com a visualização B, usando o *tablet*, e depois com o rato e o teclado. Após estes dois testes, o participante interagiu com a visualização A, começando com o *tablet*, e depois usou esta visualização com o rato e o teclado. As combinações geradas foram as seguintes:

- para o primeiro participante (P1), começou-se com a combinação ART;
- o segundo (P2) testou o sistema com a combinação BRT;
- o participante número três (P3) usou a combinação ARR;
- o quarto participante (P4) testou com a combinação BRR;
- para o quinto participante (P5) usou-se a combinação ATR;
- o sexto participante (P6) testou com a combinação BTR;
- o participante número sete (P7) utilizou o sistema com a combinação ATT;
- por fim, o oitavo participante (P8) usou a combinação BTT.

Para cada visualização foi criado um conjunto de tarefas de análise, para que os utilizadores testassem o sistema de uma forma repetível e controlada. Cada tarefa contém duas frases, que são similares, porque foi necessário criar uma tarefa que pudesse ser utilizada para testar a interação com o rato e o teclado e outra com o *tablet*. Para a visualização A foram criadas as seguintes tarefas:

- Tarefa 1:
    - Indique-me a quantidade de linhas de código da classe com o nome “Easing”.
    - Quantas linhas de código tem a classe “Tween”?
- Tarefa 2:
    - Indique-me os nomes das classes com um valor superior à “Query”.
    - Quais são as classes, com valor inferior, mais próximos do “Sort”?

A visualização B conta com as seguintes tarefas:

- Tarefa 1:
    - Procure pelo veículo Peugeot e-208 e indique-me qual é o peso dele.
    - Diga-me qual é o custo do Nissan Leaf de primeira geração.
- Tarefa 2:
    - Escolha o veículo com o menor preço possível, tendo em conta que pretende ter o maior alcance possível.
    - O que é que acontece com o custo do veículo à medida que a capacidade das suas baterias aumenta?

A avaliação dos testes foi realizada ao aplicar a escala de usabilidade do sistema (SUS) (Brooke, 1996a) para cada conjunto de tarefas que o utilizador realizou. O SUS é um conjunto de dez perguntas que são colocadas numa escala de 1 a 5, sendo que o valor 1 corresponde a uma situação de discordância total, o valor 3 representa uma opção de neutralidade (o utilizador não concorda, nem discorda) e o valor 5 indica uma concordância total com a frase enunciada. Este questionário, adaptado para a língua portuguesa, encontra-se presente na secção B.1, no Anexo B.

No início de cada sessão de testes, foi apresentado um questionário para que se pudesse fazer uma análise demográfica dos participantes deste processo de avaliação. Este questionário inclui questões tais como a idade, o género, as habilitações literárias e perguntas sobre o grau de experiência com tecnologias relevantes a este trabalho. Pode-se encontrar este questionário na subsecção B.2 do Anexo B.

4.2 Resultados

Dos oito participantes do processo de avaliação, seis eram do sexo masculino e os outros dois do sexo feminino. A idade destes participantes está representada, graficamente, na Figura 4.1.

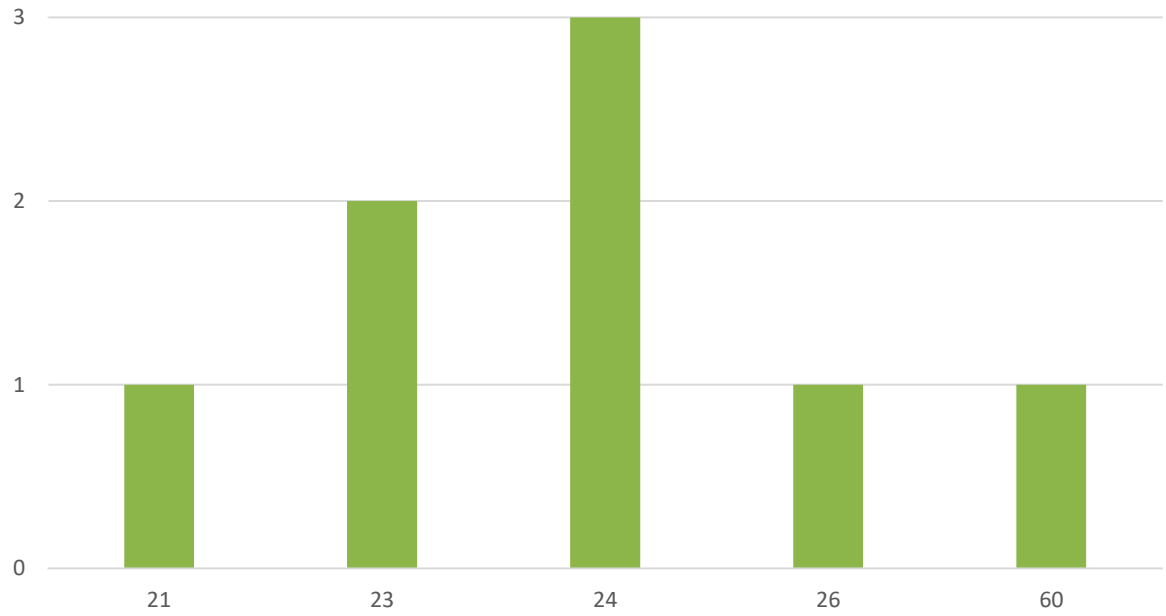
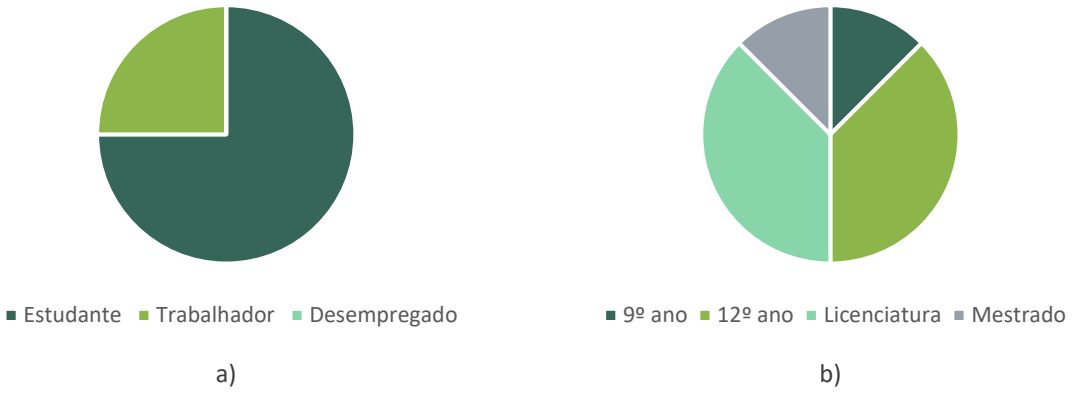


Figura 4.1 – Representação gráfica das idades dos participantes deste processo de avaliação.

A média das idades dos participantes é 28,125 anos de idade com um desvio-padrão de 12,119.

Como é possível observar na Figura 4.2 a), a maioria dos participantes deste estudo eram estudantes, sendo todos eles estudantes universitários. Dos participantes que eram trabalhadores um era trabalhador-estudante. As habilitações literárias de cada um destes participantes está representada na Figura 4.2 b).



a) b)

Figura 4.2 – Gráficos circulares contendo a informação sobre os participantes: a) situação profissional; b) habilitações académicas.

Aos participantes que a sua situação profissional era a de trabalhador, foi-lhes apresentada uma questão sobre se no seu trabalho utilizavam ferramentas de gráficos ou de visualizações de dados. Contudo, nenhum destes utiliza visualizações de dados no trabalho.

A Figura 4.3 mostra os resultados de uma autoavaliação realizada pelos utilizadores.

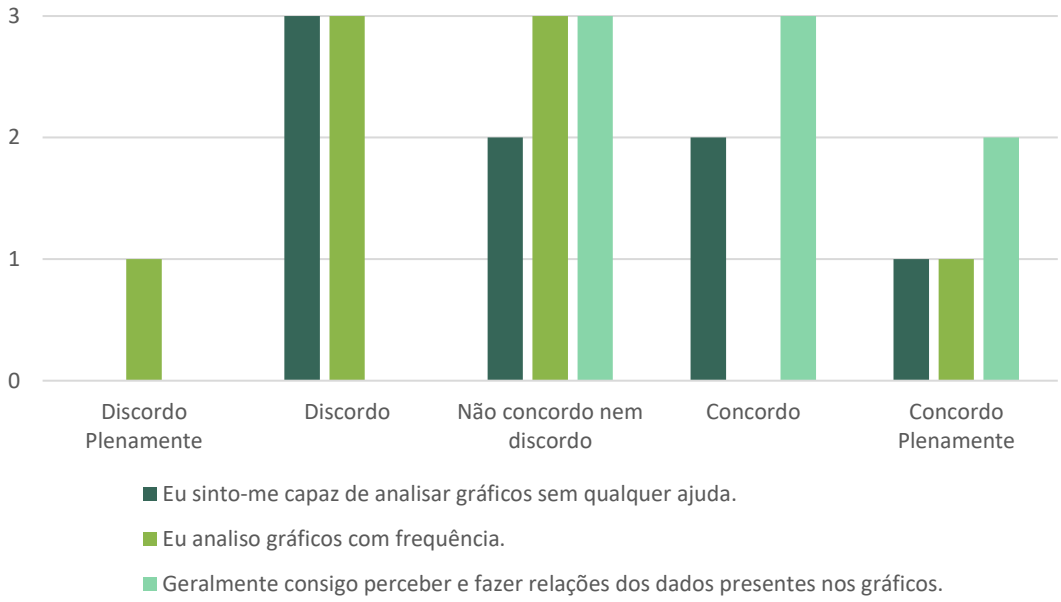


Figura 4.3 – Grau de experiência dos participantes em relação a gráficos e visualizações de dados.

Nesta autoavaliação, a grande maioria dos utilizadores indicou que não têm o hábito de analisar gráficos. Muitos destes também não se sentem muito à vontade com gráficos que não sejam os gráficos mais típicos, como por exemplo: gráficos de barras, de linhas, de colunas, circulares, entre outros.

A última observação sobre os participantes é sobre a mão dominante de cada um destes, sendo que o participante P4 era o único esquerdino e os restantes eram todos destros.

Para os resultados dos questionários SUS poderem ser úteis, é necessário calcular a sua pontuação, seguindo as indicações de Brooke. Este cálculo é feito da seguinte forma:

- às perguntas ímpares subtrai-se um valor;
- nas perguntas pares, faz-se o cálculo de cinco menos o valor atribuído à pergunta;
- soma-se todos os valores obtidos;
- e, por fim, multiplica-se por 2.5, obtendo-se o valor geral de usabilidade do sistema.

Apesar de este valor pertencer a uma escala de 0 a 100, este não deve ser visto como uma percentagem, mas sim como uma indicação de que se encontra num determinado percentil[5] (Brooke, 1996b). O resultado destes cálculos, aplicados aos questionários preenchidos pelos participantes, está presente na Tabela 4.1.

Tabela 4.1 – Resultados obtidos dos questionários SUS para cada combinação de visualização e do método de interação testado.

	SUS_AR	SUS_AT	SUS_BR	SUS_BT
Média	72.813	69.688	75.313	61.563
Desvio-padrão	15.552	20.197	12.133	20.131

É possível constatar que os testes realizados com a interação com o rato e o teclado apresentam um valor acima de 70, o que indica que a usabilidade deste sistema é boa. A interação bimanual de toque e de caneta apresenta valores inferiores, o que indica que este modelo de interação poderá ter algum problema de usabilidade.

Para que se possa ver em melhor detalhe os resultados sobre cada visualização e sobre cada tipo de interação testada, foi calculada a média combinada destes e o respetivo desvio-padrão. Estes valores encontram-se na Tabela 4.2.

Tabela 4.2 – Resultados do SUS agrupados por tipo de visualização e por método de interação.

	Visualização A	Visualização B	Interação rato	Interação *tablet*
Média	71.250	68.438	74.063	65.625
Desvio-padrão	17.488	17.556	13.536	19.927

Pode-se ver, na Figura 4.4, uma representação da dispersão dos valores obtidos, através do questionário SUS.

[5] Percentil é uma métrica de estatística que indica quantos elementos, em percentagem, têm um valor igual ou inferior ao que está a ser observado. Ou seja, o n-ésimo percentil indica que n% dos elementos analisados possuem um valor igual ou inferior ao valor que é representado por este percentil (Market Business News, sem data; Merriam-Webster, sem data).

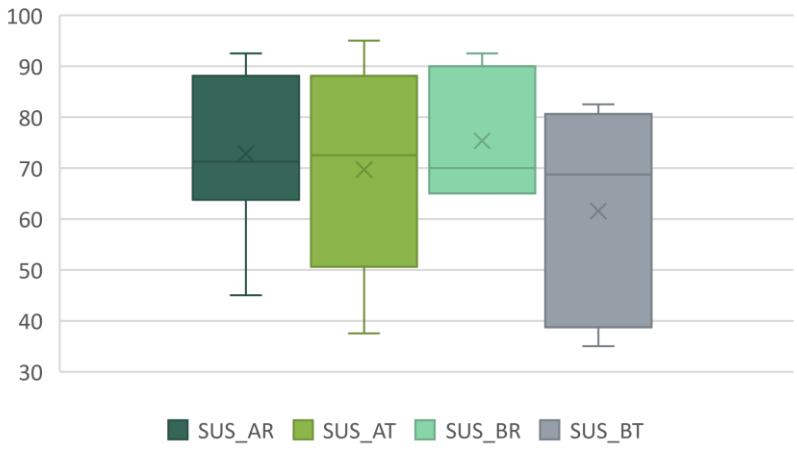


Figura 4.4 – Gráfico de caixa de bigodes dos resultados obtidos no SUS.

No gráfico representado na Figura 4.4, a linha central em cada caixa representa a média e a cruz dentro da caixa representa a mediana. Pode-se observar, nestes gráficos de caixa e bigodes, que a interação através do rato e do teclado apresenta valores menos dispersos e com valores superiores, quando comparados com os resultados dos testes com a mesma visualização, utilizando a interação bimanual de toque e de caneta.

Neste questionário de usabilidade (ver secção B.1 do Anexo B), para além das dez perguntas do SUS, foi incluída uma última questão. Nesta questão, pediu-se ao participante para classificar qualitativamente a facilidade de uso do sistema testado. Os resultados dessa pergunta encontram-se na Figura 4.5.

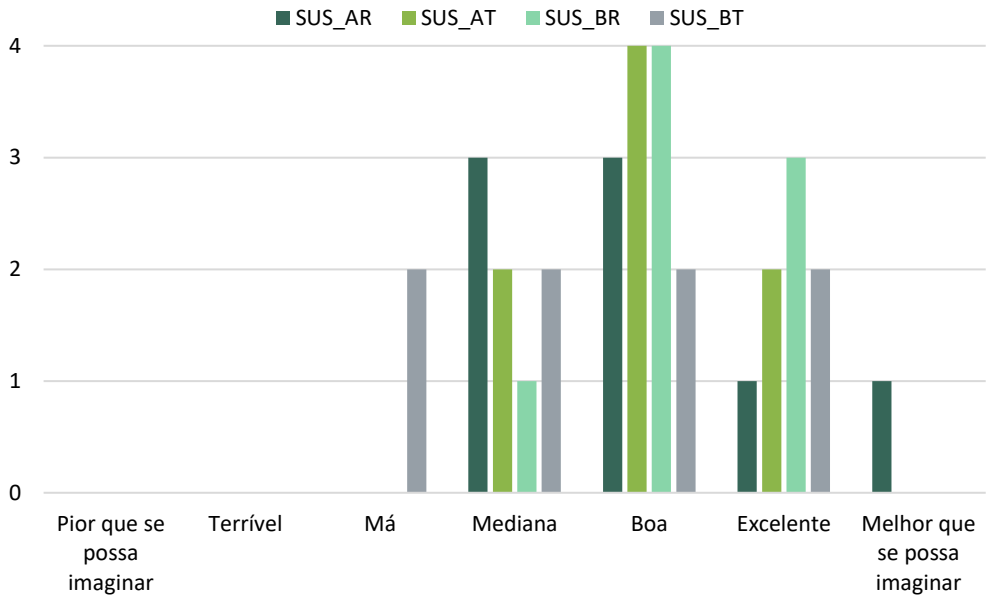


Figura 4.5 – Resultados obtidos na questão qualitativa do questionário de usabilidade SUS.

Das várias tarefas que cada participante realizou, foi-se apontando a resposta dada por cada um destes, e, no fim do processo de avaliação, calculou-se quantas respostas foram dadas corretamente.

Na Tabela 4.3 estão representadas as percentagens, em média, de respostas corretas dadas pelos utilizadores.

Tabela 4.3 – Percentagem de respostas certas agrupadas por tipo de visualização e por método de interação.

	Visualização A	Visualização B	Interação rato	Interação *tablet*
Média (%)	71.250	78.125	78.125	75.000
Desvio-padrão (%)	25.000	5.413	18.488	17.678

A percentagem de respostas corretas, por tarefa, pode ser observada na Figura 4.6, onde pode-se constatar que não houve diferença quase nenhuma na quantidade de respostas dadas, quer seja pelo tipo de visualização, quer seja pela interação utilizada na avaliação.

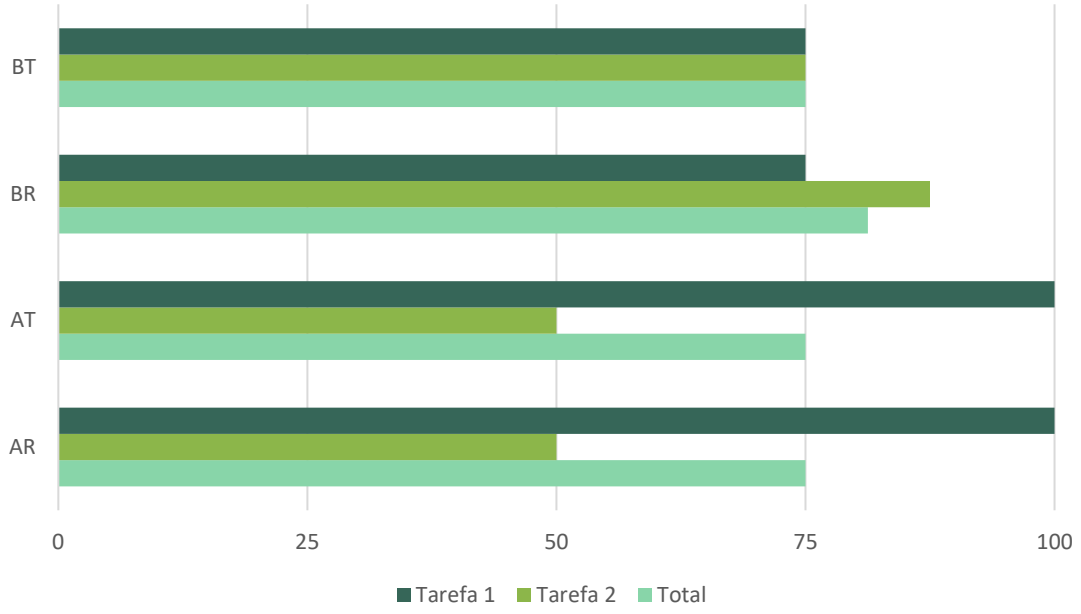


Figura 4.6 – Percentagem de respostas corretas por tarefa e o total.

Nesta avaliação também se apontou o tempo que cada participante demorou a realizar cada tarefa. Na Tabela 4.4 pode-se ver o cálculo dos tempos médios e do desvio-padrão, quando agrupados por tipo de visualização ou pelo tipo de interação utilizada.

Tabela 4.4 – Tempo médio, em segundos, que os utilizadores demoraram a realizar o teste, agrupado por tipo de visualização e por método de interação.

	Visualização A	Visualização B	Interação rato	Interação *tablet*
Média (s)	63.250	93.000	61.688	94.562
Desvio-padrão (s)	53.512	65.823	35.695	76.316

Na Tabela 4.4 é possível constatar que os utilizadores demoraram menos tempo, em média, na visualização A do que na B e que o mesmo acontece com a interação com o rato e com o teclado, quando comparada com a interação bimanual de toque e de caneta.

A informação detalhada sobre o tempo médio que cada tarefa demorou a ser executada está representada na Figura 4.7.

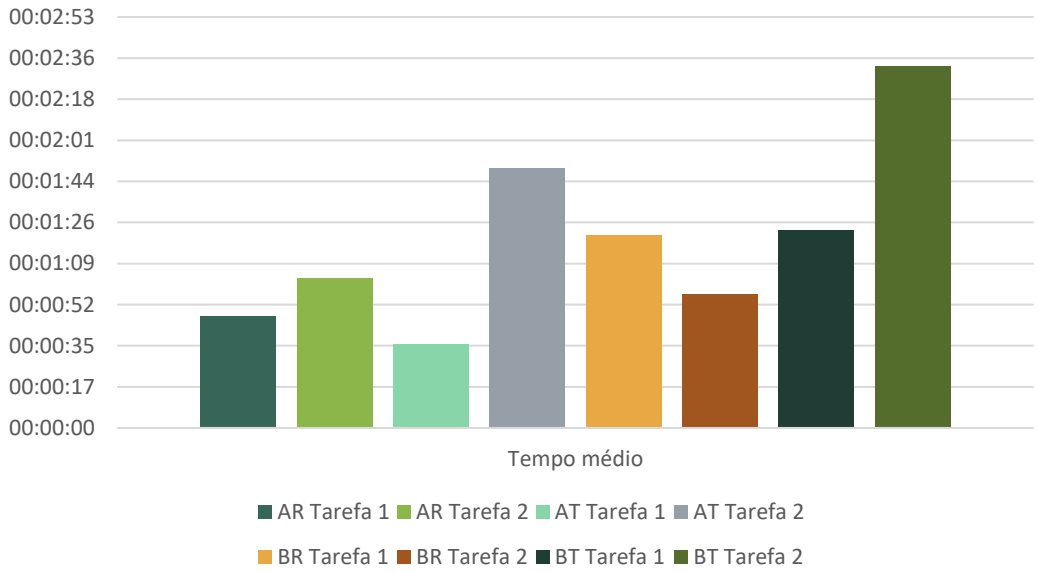


Figura 4.7 – Tempo médio de execução de cada tarefa.

Na Figura 4.7 verifica-se que a interação bimanual levou, em média, menos tempo a ser realizada do que com a interação com o rato e o teclado, o que contraria a tendência de as tarefas realizadas com a interação de toque e de caneta demorarem mais tempo a serem concluídas.

4.3 Discussão

Ao se analisar os resultados obtidos, pode-se observar que existe uma tendência de os utilizadores classificarem a usabilidade do sistema com um valor mais elevado nos casos em

que são utilizados o rato e o teclado, quando comparado com a interação de toque e de caneta, como está evidenciado na Figura 4.8.

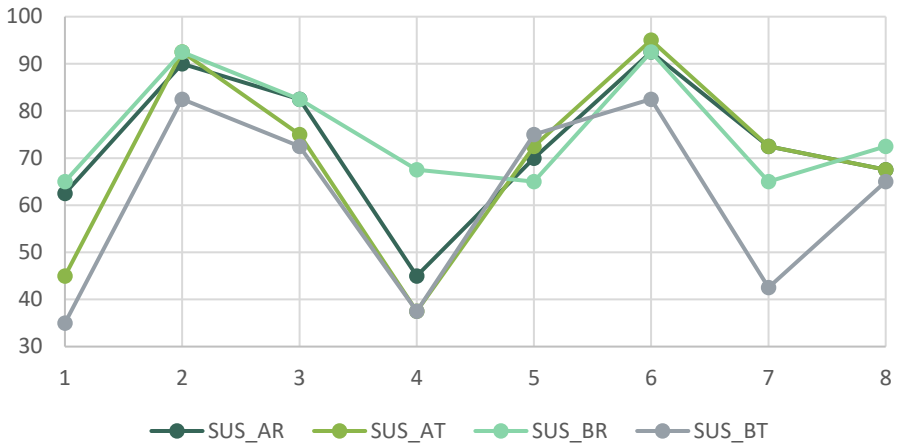


Figura 4.8 – Gráfico de linhas das pontuações do questionário SUS calculadas para cada participante.

Nesta figura também pode-se observar que alguns utilizadores tiveram dificuldades em utilizar o sistema quer fosse com o rato e o teclado ou com as interações de toque e de caneta. Estas reduções da pontuação da usabilidade do sistema devem-se, em parte, à dificuldade sentida por alguns destes participantes em analisar os gráficos. Estas dificuldades surgem porque os utilizadores não estão habituados a analisar gráficos em que a informação não esteja toda visível, por outras palavras, estes utilizadores estão habituados a analisar um gráfico apenas pela informação que lhes é apresentada imediatamente e não tentam interagir muito com este.

Os participantes indicaram, através do questionário, que as interações de rato e de teclado são melhores que as de toque e de caneta. Estes participantes, quando questionados, informalmente, sobre qual destes métodos de interação preferiram os participantes indicaram, na sua maioria, que preferiram a interação com o rato e o teclado por ser aquela a que estão mais habituados, no entanto, houve alguns que disseram que poderiam se habituar a usar o *tablet* e a interação de toque e de caneta. Indicaram também que, apesar de ter sido mais fácil de utilizar o rato e o teclado para analisar gráficos, depois de já estarem habituados à interação bimanual talvez viessem a preferir este modo de interação.

Desta análise dos resultados obtidos pode-se concluir que o gráfico de bolhas (visualização A) tem uma melhor usabilidade do que o conjunto do gráfico de dispersão e de radar (visualização B). Este resultado deve-se, em parte, ao facto de esta segunda visualização ser

mais complexa que a primeira e de ser necessário realizar mais operações sobre o gráfico, para poder usá-la com todo o seu potencial.

Ao utilizarem a visualização A, houve dois participantes que ao realizarem a tarefa utilizaram a combinação de teclas "CTRL+F". Esta combinação de teclas abre uma caixa de pesquisa, na qual estes utilizadores inseriram o nome da classe que estavam à procura. Este atalho apenas é permitido na interação com o rato e o teclado porque o navegador *web* utilizado para este método de interação tem este atalho disponível. Na interação de toque e de caneta os utilizadores apenas podiam fazer uma pesquisa visual dos nomes das classes.

Em relação à visualização B houve uma certa resistência por parte de alguns participantes em utilizarem o gráfico de radar. Estes utilizadores preferiram utilizar o gráfico de dispersão para pesquisarem informação, mesmo quando não podiam observar os valores exatos. Houve um participante que, ao usar esta visualização, teve dificuldade em se adaptar ao pequeno desfasamento que ocorre entre a ponta da caneta e o sítio onde esta está realmente a pressionar. Esta inadaptação levou a que este participante se sentisse frustrado com o sistema, uma vez que não estava a conseguir identificar o problema que fazia com que este não conseguisse selecionar os pontos no gráfico. O mesmo utilizador também teve problemas com a interação em que se utiliza três dedos e inclina-se a caneta, porque estava a tentar executá-la sobre o gráfico de radar e com a sua mão esquerda a obstruir a visualização. Esta obstrução de visão impediu-o de se aperceber que realmente as suas ações estavam a ter efeito na visualização. O participante P4, que é o único participante esquerdino, revelou que nesta visualização teve alguns problemas em visualizar a informação apresentada porque o gráfico de dispersão ficava coberto pela mão devido à maneira como este coloca o braço e a mão sobre o *tablet* ao usar a caneta.

Na Figura 4.9 estão presentes os diagramas de caixas e bigodes com a informação do tempo utilizado para concluir cada tarefa. Nesta figura pode-se notar a existência de alguns extremos (*outliers*) nalgumas tarefas, representados por bolas acima das caixas de bigodes. Estes extremos revelam que alguns utilizadores demoraram muito mais tempo que os restantes utilizadores.

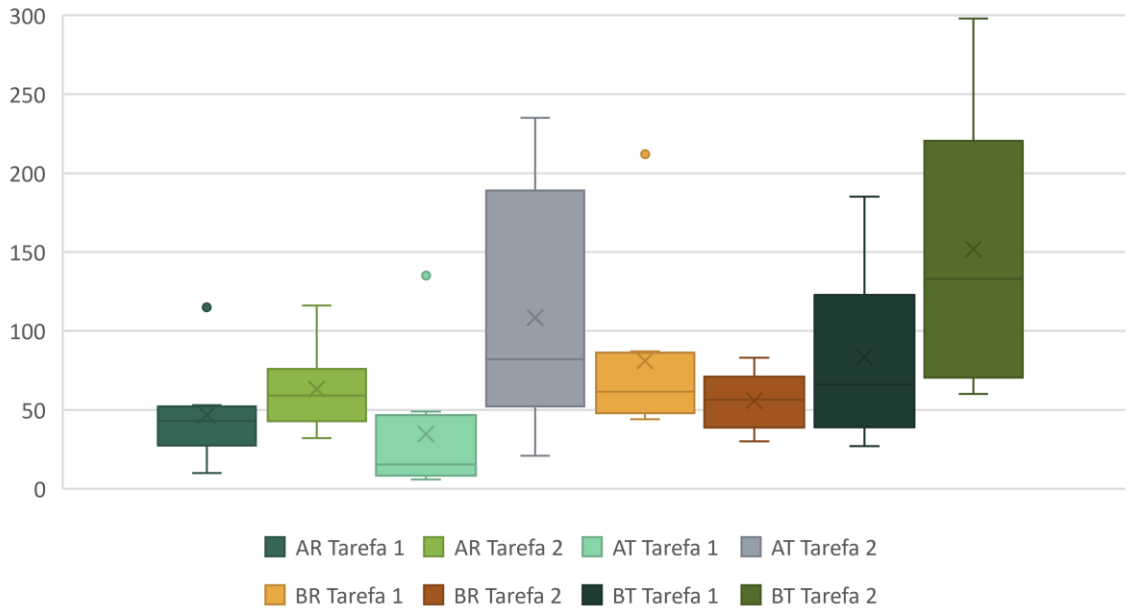


Figura 4.9 – Gráfico de caixa de bigodes do tempo utilizado para realizar as tarefas.

Com esta figura e a Figura 4.7 pode-se concluir que os participantes demoram mais tempo a realizar as tarefas quando estão a usar a interação de toque e de caneta do que com o rato e o teclado. Uma das razões que pode causar este aumento no tempo para realizar as tarefas pode ser devido à maior familiaridade dos utilizadores com a interação com o rato e o teclado, uma vez que maioria das pessoas ainda utiliza este método de interação como o principal método de interação com os computadores. Outra razão passa pela maior complexidade dos gráficos e dos próprios dados presentes na visualização B.

5 Conclusões

Esta dissertação teve como objetivo a criação de um sistema em que se pudesse detetar as entradas de toque e de caneta num *tablet* gráfico Wacom. Através desta interação realizada pelo utilizador, o sistema permite que haja uma componente de interação entre as visualizações de dados e as ações de interação realizadas. Para este propósito, foi criado um sistema em que fosse possível extrair os dados sobre os dedos e a caneta em relação ao *tablet*. Estes dados foram extraídos ao utilizar duas API de acesso às informações do *tablet*, nomeadamente, a bbTablet e a Wacom Feel Multi-Touch SDK. O acesso a estas foi feito através de uma aplicação desenvolvida, tendo por base o openFrameworks.

Nesta aplicação foi implementada uma classe com métodos de acesso, de armazenamento, de conversão e de distribuição dos dados lidos. Na aplicação de openFrameworks, instalou-se o *plugin* ofxAwesomiumPlus, que permitiu adicionar o navegador *web* para OpenGL, da Awesomium, na janela de renderização do openFrameworks. Foi através deste navegador que se acedeu às visualizações de dados implementadas, através da utilização da biblioteca de visualização D3.js. No total, foram feitas três visualizações de dados com interação bimanual de toque e de caneta. Para enviar os dados provenientes do *tablet* para o navegador, foi necessário criar métodos para converter os dados em bruto, vindos do *tablet*, em eventos de pressão numa dada tecla do teclado ou em eventos de rato.

Foram realizados testes com utilizadores para que se pudesse determinar se o modelo de interação bimanual de toque e de caneta era viável no contexto de visualização de dados. Destes testes realizados na fase de avaliação do protótipo, pode-se concluir que a modalidade de interação bimanual de toque e de caneta apresenta alguns obstáculos, nomeadamente em termos de usabilidade e de tempo necessário para realizar as tarefas. Estes obstáculos aparentam dever-se a alguma falta de experiência, por parte dos utilizadores, em analisar gráficos para além dos mais conhecidos e devido à falta de familiaridade em utilizar sistemas de interação bimanual. Apesar disto, os resultados indicam que não existe uma diminuição na percentagem de respostas corretas. Com estes resultados, conclui-se que apesar de as interações de toque e de caneta, geralmente, terem uma maior precisão e rapidez na realização de tarefas e serem mais intuitivas (Brandl et al., 2010), no contexto de visualização

de dados, este tipo de interação parece sobrecarregar os utilizadores. Esta sobrecarga pode ser justificada pelo facto dos utilizadores já se encontrarem muito concentrados com a tarefa de analisar os gráficos e, ao adicionar as interações bimanuais, está-se efetivamente a aumentar a carga cognitiva dos utilizadores, devido à utilização de ambas as mãos para as interações.

5.1 Limitações

Nesta secção são apresentadas algumas das limitações que foram encontradas no desenvolvimento deste trabalho.

Não é possível utilizar gestos com a caneta sobre as visualizações de dados, para além do gesto de inclinar a caneta, porque não é possível enviar as posições do rato ou as formas realizadas para o navegador embutido no ofxAwesomiumPlus.

A escolha da biblioteca de visualizações D3.js restringe o tipo de visualização a apenas visualizações bidimensionais e força a utilização de um servidor *web*, aumentando a quantidade de memória utlizada por este protótipo e a sua complexidade.

5.2 Trabalho futuro

Nesta subsecção serão apresentadas possíveis alterações que poderão vir a ser realizadas para que se possa melhorar o sistema implementado. Estas alterações podem ser em termos de interações disponíveis ao utilizador, de mudanças de *frameworks* ou de *software* utilizados e possíveis alterações que não alteram as funcionalidades do sistema, mas que permitem uma maior extensibilidade do sistema desenvolvido.

Os testes realizados durante a fase de avaliação do sistema poderiam ser refeitos com mais utilizadores para verificar se as tendências encontradas continuariam a existir. Também podia-se realizar algumas alterações na metodologia, para que se possa extrair informação sobre o número de ações e de erros ou como é que os utilizadores usam o sistema. Estas alterações iriam permitir retirar mais informações da análise de sistema realizada.

Uma das mudanças a fazer, no futuro, seria mudar a *framework* do Awesomium (Awesomium e Simmons, 2017), porque, neste momento, esta já se encontra deprecada, ou

seja, esta *framework* já não tem qualquer desenvolvimento a ser feito. Uma *framework* que poderia substituir esta seria a ultralight (Simmons, 2019), que é desenvolvida pela mesma pessoa que desenvolvia a Awesomium originalmente. Esta *framework* foi desenvolvida para substituir a Awesomium, uma vez que a Awesomium já se encontrava um bocado desatualizada e porque pretenderam mudar o motor de renderização de páginas *web* para um mais recente e mais leve. Apesar da ultralight já estar disponível durante o desenvolvimento deste trabalho, esta ainda não se encontrava numa versão matura e estável, impedindo, assim, a sua utilização no protótipo. Outro fator, que impediu a utilização do ultralight foi a inexistência de um *plugin* para openFrameworks, na altura em que o protótipo já estava a começar a ser desenvolvido. O primeiro *plugin* para esta *framework* surgiu em agosto de 2019 (Byun, 2019).

A principal alteração ao código desenvolvido seria criar um ficheiro JavaScript com todos os métodos de interpretação de eventos de teclado. Este ficheiro com todos métodos de interpretação permitiria evitar a repetição de código e também simplificaria a programação destas funcionalidades.

Uma das possíveis adições às funcionalidades deste sistema, seria a criação de um método de seleção baseado numa ferramenta de laço (*lasso tool*). Com este método de seleção, o utilizador desenharia uma forma, em torno dos elementos de dados que este quisesse selecionar, e depois o sistema iria interpretá-la e marcar como selecionados todos os elementos que estivessem no interior da forma desenhada.

Poder-se-ia mudar de biblioteca de visualização, com o objetivo de se vir a ter-interações mais naturais e fluidas. Esta mudança podia ser feita para uma biblioteca em que não houvesse uma camada intermédia, o que leva à perda de dados ou à perda de precisão, e que a biblioteca permitisse que se passasse os dados em bruto para a visualização. Uma alternativa seria uma biblioteca de visualização similar à VTK, mas que fosse mais simples de estender as funcionalidades desta.

6 Referências bibliográficas

amCharts, 2020. Rotating globe - amCharts [WWW Document]. URL https://www.amcharts.com/demos/rotating-globe/ (acedido 1.17.20).

amCharts, 2019. amCharts - JavaScript charts & maps [WWW Document]. URL https://www.amcharts.com/ (acedido 10.8.19).

AnyChart, 2019. AnyChart - JavaScript charting library [WWW Document]. URL https://www.anychart.com/ (acedido 10.8.19).

Apple, Inc., 2020a. Handling Input from Apple Pencil | Apple Developer Documentation [WWW Document]. URL https://developer.apple.com/documentation/uikit/pencil_interactions/handling_input_from_apple_pencil (acedido 2.7.20).

Apple, Inc., 2020b. UITouch | Apple Developer Documentation [WWW Document]. URL https://developer.apple.com/documentation/uikit/uitouch#declaration (acedido 2.7.20).

Apple, Inc., 2019a. iPad Pro - Technical Specifications - Apple [WWW Document]. URL https://www.apple.com/ipad-pro/specs/ (acedido 8.22.19).

Apple, Inc., 2019b. Apple Pencil - Apple [WWW Document]. URL https://www.apple.com/apple-pencil/ (acedido 8.22.19).

Awesomium, Simmons, A., 2017. Awesomium - HTML UI Engine [WWW Document]. URL https://web.archive.org/web/20170608212207/http://www.awesomium.com/ (acedido 6.5.19).

Baxter, W., 2009. bbTablet [WWW Document]. URL http://www.billbaxter.com/projects/bbtablet/ (acedido 4.10.19).

Beck, F., Stumpe, B., 1973. Two devices for operator interaction in the central control of the new CERN accelerator. CERN, Geneva. https://doi.org/10.5170/CERN-1973-006

Bostock, M., 2019a. D3.js (Data-Driven Documents) [WWW Document]. URL https://d3js.org/

(acedido 5.2.19).

Bostock, M., 2019b. U.S. Population by State, 1790–1990 [WWW Document]. URL https://observablehq.com/@mbostock/u-s-population-by-state-1790-1990 (acedido 8.14.19).

Bostock, M., 2017. Radial Tidy Tree [WWW Document]. URL https://observablehq.com/@d3/radial-tidy-tree (acedido 8.9.19).

Brandl, P., Haller, M., Forlines, C., Wigdor, D., Shen, C., 2010. Combining and measuring the benefits of bimanual pen and direct-touch interaction on horizontal interfaces, em: Proceedings of the working conference on Advanced visual interfaces - AVI '08. ACM Press, New York, New York, USA, pp. 154–161. https://doi.org/10.1145/1385569.1385595

Brooke, J., 1996a. SUS: A quick and dirty usability scale, em: Jordan, P.W., Thomas, B., McClelland, I.L., Weerdmeester, B.A. (Eds.), Usability evaluation in industry. Taylor & Francis, London, pp. 189–194. https://doi.org/https://doi.org/10.1201/9781498710411

Brooke, J., 1996b. SUS: A quick and dirty usability scale. Usability Eval. Ind. 189–194.

Büring, T., Gerken, J., Reiterer, H., 2006. User interaction with scatterplots on small screens - A comparative evaluation of geometric-semantic zoom and fisheye distortion, em: IEEE Transactions on Visualization and Computer Graphics. pp. 829–836. https://doi.org/10.1109/TVCG.2006.187

Buxton, B., 2007. Multi-Touch Systems that I Have Known and Loved [WWW Document]. Microsoft Res. URL http://www.billbuxton.com/multitouchOverview.html (acedido 9.6.19).

Byun, I., 2019. ofxUltralight-byun [WWW Document]. URL https://github.com/IanByun/ofxUltralight-byun (acedido 1.7.20).

Cesanta Software Limited, Lyubka, S., 2019. Mongoose [WWW Document]. URL https://github.com/cesanta/mongoose (acedido 4.10.19).

Colwell Jr., W.C., Hurst, G.S., 1975. US patent #3,911,215: Discriminating contact sensor. US3911215A.

Dimara, E., Perin, C., 2020. What is Interaction for Data Visualization? IEEE Trans. Vis. Comput.

Graph. 26, 119–129. https://doi.org/10.1109/TVCG.2019.2934283

DOM Standard [WWW Document], sem data. URL https://dom.spec.whatwg.org/ (acedido 2.9.20).

Guiard, Y., 1987. Asymmetric division of labor in human skilled bimanual action. J. Mot. Behav. 19, 486–517. https://doi.org/10.1080/00222895.1987.10735426

Hamilton, W., Kerne, A., Robbins, T., 2012. High-performance pen + touch modality interactions: A real-time strategy game ESports context, em: Proceedings of the 25th Annual ACM Symposium on User Interface Software and Technology, UIST '12. Association for Computing Machinery, Cambridge, Massachusetts, USA, pp. 309–318. https://doi.org/10.1145/2380116.2380156

Ilin, R., 2016. ofxAwesomiumPlus [WWW Document]. URL http://razvanilin.com/ofxawesomiumplus/ (acedido 4.10.19).

Isenberg, P., Isenberg, T., von Zadow, U., Hesselmann, T., Tang, A., 2013. Data Visualization on Interactive Surfaces: A Research Agenda. IEEE Comput. Graph. Appl. 33, 16–24. https://doi.org/10.1109/mcg.2013.24

Jo, J., L'Yi, S., Lee, B., Seo, J., 2017. TouchPivot: Blending WIMP & Post-WIMP interfaces for data exploration on tablet devices, em: Conference on Human Factors in Computing Systems - Proceedings. Association for Computing Machinery, pp. 2660–2671. https://doi.org/10.1145/3025453.3025752

Johnson, E.A., 1967. Touch displays: A programmed man-machine interface. Ergonomics 10, 271–277. https://doi.org/10.1080/00140136708930868

Johnson, E.A., 1965. Touch display—a novel input/output device for computers. Electron. Lett. 1, 219–220. https://doi.org/10.1049/el:19650200

Johnson, I., 2016. Served [WWW Document]. URL https://github.com/enjalot/served (acedido 5.2.19).

Karat, J., McDonald, J.E., Anderson, M., 1986. A comparison of menu selection techniques: touch panel, mouse and keyboard. Int. J. Man. Mach. Stud. 25, 73–88. https://doi.org/10.1016/S0020-7373(86)80034-7

Kasday, L.R., 1984. US patent #4,484,179: Touch position sensitive surface. US4484179A.

Keefe, D.F., 2010. Integrating visualization and interaction research to improve scientific workflows. IEEE Comput. Graph. Appl. 30, 8–13. https://doi.org/10.1109/MCG.2010.30

Kennedy, A.B.W., Sankey, H.R., 1898. The Thermal Efficiency of Steam Engines. Report of the Committee Appointed to the Council Upon the Subject of the Definition of a Standard or Standards of Thermal Efficiency for Steam Engines: With an Introductory Note. (Including Appendixes and Plate at. Minutes Proc. Inst. Civ. Eng. 134, 278–312. https://doi.org/10.1680/imotp.1898.19100

Kim, N.W., Pfister, H., Henry Riche, N., Bach, B., Xu, G., Brehmer, M., Hinckley, K., Pahud, M., Xia, H., McGuffin, M.J., 2019. DataToon, em: Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems - CHI '19. ACM Press, New York, New York, USA, pp. 1–12. https://doi.org/10.1145/3290605.3300335

Kin, K., Agrawala, M., DeRose, T., 2009. Determining the Benefits of Direct-Touch, Bimanual, and Multifinger Input on a Multitouch Workstation, em: Proceedings of the Graphics Interface Conference (GI'09). Canadian Information Processing Society, Kelowna, British Columbia, Canada, pp. 119–124.

Kitware, Inc., 2019. VTK - The Visualization Toolkit [WWW Document]. URL https://vtk.org/ (acedido 6.5.19).

Kitware, Inc., 2014. Paraview/VTK Airflow Example [WWW Document]. URL https://www.paraview.org/wp-content/uploads/2014/04/full_F35-3.png (acedido 8.5.19).

Kitware, Inc., sem data. VTK Medical Example [WWW Document]. URL https://vtk.org/Wiki/images/8/82/VTK_Examples_Baseline_Medical_TestMedicalDemo2.png (acedido 8.7.19).

Lee, B., Isenberg, P., Riche, N.H., Carpendale, S., 2012. Beyond mouse and keyboard: Expanding design considerations for information visualization interactions. IEEE Trans. Vis. Comput. Graph. 18, 2689–2698. https://doi.org/10.1109/TVCG.2012.204

Mack, R., Lang, K., 1989. A Benchmark Comparison of Mouse and Touch Interface Techniques for an Intelligent Workstation Windowing Environment. Proc. Hum. Factors Soc. Annu. Meet. 33, 325–329. https://doi.org/10.1177/154193128903300520

Mack, R., Montaniz, F., 1991. A Comparison of Touch and Mouse Interaction Techniques for a Graphical Windowing Software Environment. Proc. Hum. Factors Soc. Annu. Meet. 35, 286–289. https://doi.org/10.1177/154193129103500510

Malvern Radar and Technology History Society, Johnson, E.A., 2016. 1965 – The Touchscreen [WWW Document]. MRATHS. URL https://mraths.org.uk/?page_id=531 (acedido 8.22.19).

Market Business News, sem data. What is percentile? Definition and meaning [WWW Document]. URL https://marketbusinessnews.com/financial-glossary/percentile-definition-meaning/ (acedido 1.31.20).

Mason, B., 2019. Why scientists need to be better at data visualization. Knowable Mag. https://doi.org/10.1146/knowable-110919-1

Merriam-Webster, I., sem data. Percentile | Definition of Percentile by Merriam-Webster [WWW Document]. URL https://www.merriam-webster.com/dictionary/percentile (acedido 1.31.20).

Microsoft Corporation, 2019a. Microsoft Surface Pro 6 – Technical Specifications – Microsoft Surface [WWW Document]. URL https://www.microsoft.com/en-us/surface/devices/surface-pro-6/tech-specs (acedido 8.22.19).

Microsoft Corporation, 2019b. Visual Studio IDE [WWW Document]. URL https://visualstudio.microsoft.com/vs/ (acedido 1.23.20).

Moeller, J., Kerne, A., 2012. ZeroTouch: An optical multi-touch and free-air interaction architecture, em: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. Association for Computing Machinery, Austin, Texas, USA, pp. 2165–2174. https://doi.org/10.1145/2207676.2208368

openFrameworks Community, 2020. ofBaseApp [WWW Document]. URL https://openframeworks.cc/documentation/application/ofBaseApp/#show_setup (acedido 1.14.20).

openFrameworks Community, 2019. openFrameworks [WWW Document]. URL https://openframeworks.cc/ (acedido 4.10.19).

Romat, H., Riche, N.H., Hinckley, K., Lee, B., Appert, C., Pietriga, E., Collins, C., 2019. Activeink:

(th)inking with data, em: Conference on Human Factors in Computing Systems - Proceedings. ACM Press, New York, New York, USA, pp. 1–13. https://doi.org/10.1145/3290605.3300272

Sadana, R., Stasko, J., 2014. Designing and implementing an interactive scatterplot visualization for a tablet computer, em: Proceedings of the 2014 International Working Conference on Advanced Visual Interfaces - AVI '14. ACM Press, New York, New York, USA, pp. 265–272. https://doi.org/10.1145/2598153.2598163

Samsung, 2017. How the Notebook 9 Pen Blends Powerful Performance with Premium Design – Samsung Global Newsroom [WWW Document]. URL https://news.samsung.com/global/how-the-notebook-9-pen-blends-powerful-performance-with-premium-design (acedido 8.22.19).

Simmons, A., 2019. Ultralight - HTML UI Engine [WWW Document]. URL https://ultralig.ht/ (acedido 1.7.20).

Stumpe, B., 1977. A new principle for an X-Y touch screen. Geneva.

Tang, J., Liu, J., Zhang, M., Mei, Q., 2016. Visualizing large-scale and high-dimensional data, em: Proceedings of the 25th International Conference on World Wide Web - WWW '16. ACM Press, Republic and Canton of Geneva, CHE, pp. 287–297. https://doi.org/10.1145/2872427.2883041

Wacom Co., Ltd., 2019a. Wacom Cintiq Pro 13: creative pen display | Wacom [WWW Document]. URL https://www.wacom.com/en-gb/products/pen-displays/wacom-cintiq-pro-13 (acedido 8.22.19).

Wacom Co., Ltd., 2019b. Wacom MobileStudio Pro 13 | Wacom [WWW Document]. URL https://www.wacom.com/en-gb/products/pen-computers/wacom-mobilestudio-pro-13 (acedido 8.22.19).

Wacom Co., Ltd., 2015a. Cintiq 13 HD Graphic Pen Tablet for Drawing | Wacom [WWW Document]. URL https://www.wacom.com/en/products/pen-displays/cintiq-13-hd (acedido 8.22.19).

Wacom Co., Ltd., 2015b. Wacom Cintiq 13HD Interactive Pen Display [WWW Document]. URL http://www.wacombangladesh.com/wp-content/uploads/2017/04/Wacom-Cintiq-

13HD-Interactive-Pen-Display.jpg (acedido 9.16.19).

Wacom Co., L., 2019. Wacom Feel Multi-Touch API [WWW Document]. URL https://developer-docs.wacom.com/pages/viewpage.action?pageId=10422372 (acedido 4.10.19).

Walny, J., Lee, B., Johns, P., Henry Riche, N., Carpendale, S., 2012. Understanding pen and touch interaction for data exploration on interactive whiteboards. IEEE Trans. Vis. Comput. Graph. 18, 2779–2788. https://doi.org/10.1109/TVCG.2012.275

Yee, K.-P., 2004. Two-handed interaction on a tablet display, em: Extended abstracts of the 2004 conference on Human factors and computing systems - CHI '04. ACM Press, New York, New York, USA, p. 1493. https://doi.org/10.1145/985921.986098

Zgraggen, E., Zeleznik, R., Drucker, S.M., 2014. PanoramicData: Data analysis through pen touch. IEEE Trans. Vis. Comput. Graph. 20, 2112–2121. https://doi.org/10.1109/TVCG.2014.2346293

Anexo A – Exemplos de utilização da VTK

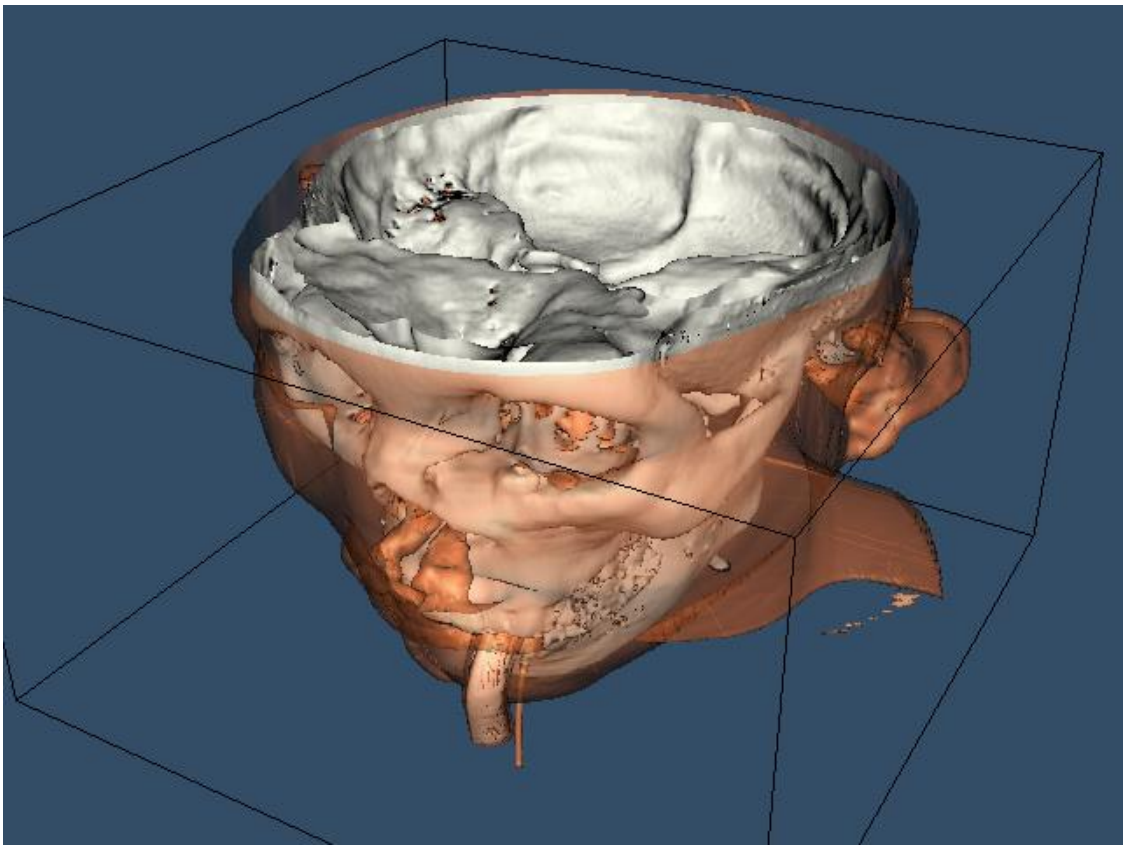


Figura A.1 – Exemplo de uma aplicação médica do *software* VTK (Kitware, Inc., sem data).

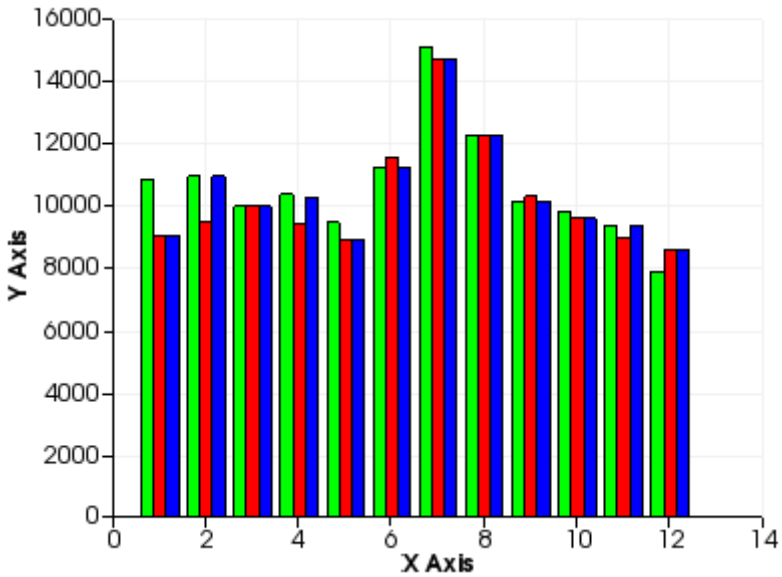


Figura A.2 – Exemplo de um gráfico de barras criado através do *software* VTK.

Anexo B – Questionários utilizados na avaliação

B.1 Escala de usabilidade do sistema (SUS)

Participante: _____ Teste: _____

Por favor marque a caixa que corresponda à sua resposta imediata a cada afirmação apresentada. Não pense durante muito tempo sobre cada afirmação. Por favor responda a todas as afirmações. Se não souber responder marque a opção "3".

	Discordo totalmente	Discordo	Não concordo nem discordo	Concordo	Concordo totalmente
1. Eu acho que gostaria de utilizar este sistema frequentemente.	1	2	3	4	5
2. Eu achei o sistema desnecessariamente complexo.	1	2	3	4	5
3. Eu acho que o sistema é fácil de utilizar.	1	2	3	4	5
4. Eu penso que precisaria de ajuda de alguém com conhecimento técnico para conseguir usar este sistema.	1	2	3	4	5
5. Eu acho que as várias funcionalidades do sistema estavam bem integradas.	1	2	3	4	5
6. Eu achei que existe demasiada inconsistência neste sistema.	1	2	3	4	5
7. Eu consigo imaginar que maioria das pessoas iria conseguir aprender a usar este sistema rapidamente.	1	2	3	4	5
8. Eu achei a utilização do sistema bastante incómoda.	1	2	3	4	5
9. Eu senti-me confiante ao utilizar o sistema.	1	2	3	4	5
10. Eu precisei de aprender muitas coisas antes que eu pudesse começar a usar o sistema.	1	2	3	4	5

Nesta última pergunta marque apenas um quadrado, consoante a sua opinião em relação à frase apresentada.

11. No geral, eu pontuaria a facilidade de uso deste sistema como:

Pior que se possa imaginar	Terrível	Má	Mediana	Boa	Excelente	Melhor que se possa imaginar

B.2 Questionário pós-teste

Questionário

Todas as respostas que fornecer neste formulário são anónimas e apenas serão usadas para a avaliação do protótipo que acabou de testar.

*Required

1. **Género** *
   Mark only one oval.
   - ◯ Feminino
   - ◯ Masculino
   - ◯ Prefiro não dizer

2. **Idade** *

3. **Habilitações académicas** *
   Mark only one oval.
   - ◯ Sem escolaridade
   - ◯ 4.º ano
   - ◯ 6.º ano
   - ◯ 9.º ano
   - ◯ 12.º ano
   - ◯ Bacharelato
   - ◯ Licenciatura
   - ◯ Mestrado
   - ◯ Doutoramento

4. **Situação profissional** *
   Mark only one oval.
   - ◯ Trabalhador *Skip to question 5.*
   - ◯ Estudante *Skip to question 8.*
   - ◯ Desempregado *Skip to question 8.*

Skip to question 6.

Sobre a sua profissão

5. **No seu trabalho utiliza alguma ferramenta de visualização ou que apresente gráficos?** *
   Mark only one oval.

   - ◯ Sim
   - ◯ Não *Skip to question 8.*

Experiência com visualizações de dados

6. **Algumas vez usou algum programa de manipulação ou de criação de gráficos?** *
   Mark only one oval.

   - ◯ Sim
   - ◯ Não
   - ◯ Talvez

7. **Selecione a opção que achar mais correta consoante a afirmação apresentada** *
   Mark only one oval per row.

	Discordo plenamente	Discordo	Não concordo nem discordo	Concordo	Concordo plenamente
Eu sinto-me capaz de analisar gráficos sem qualquer ajuda.	◯	◯	◯	◯	◯
Eu analiso gráficos com frequência.	◯	◯	◯	◯	◯
Geralmente consigo perceber e fazer relações dos dados presentes nos gráficos	◯	◯	◯	◯	◯

Experiência com dispositivos eletrónicos

8. **Possui algum destes dispositivos?** *
   Tick all that apply.

   - ☐ Computador
   - ☐ Smartphone
   - ☐ Tablet

9. **Selecione a opção que achar mais correta consoante a afirmação apresentada** *
Mark only one oval per row.

	Discordo plenamente	Discordo	Não concordo nem discordo	Concordo	Concordo plenamente
Eu uso frequentemente computadores.					
Sinto-me capaz de utilizar um computador sem ajuda de terceiros					
Eu uso frequentemente dispositivos com ecrãs táteis					
Sinto-me capaz de utilizar um dispositivo com ecrã tátil sem necessitar de ajuda					