Capítulo 6

Interagir para Perscrutar

No contexto de análise visual, interações com representações visuais dos dados é essencial, pois, de acordo com Colin Ware[1], *a good visualization is something that allows us to drill down and find more data about anything that seems important.* Diferente dos aplicativos gráficos convencionais que seguem um fluxo de controle de mão-única como o do fluxo de renderização de *OpenGL* (Fig. 5.1), visualização para perscrutação deve permitir que o usuário interfira no controle de fluxo destes sistemas de forma iterativa, compartilhando com o computador a responsabilidade pelas imagens expostas numa tela do computador, como ilustra o **ciclo de interação** na Figura 6.1.

Um cenário deste compartilhamento seria que, através dos **dispositivos de saída**, o sistema computacional exibe para usuário o estado dos dados. O usuário vê/lê a informação apresentada, interpreta, avalia e intervém no fluxo de controle através dos **dispositivos de entrada**. Os sinais de dispositivos de entrada são processados como **eventos** pelo processador que dispara um tratador apropriado para modificar o estado dos dados armazenados em memória. Finalmente, o novo estado dos dados deve ser informado para o usuário decidir a próxima interação. A iteração é repetida até que o estado de detalhes almejado seja alcançado.

De acordo com a complexidade de conhecimentos demandada, Ware[1] distingue três classes de laços (iterativos) de interação: laço de manipulação de dados, laço de exploração e navegação, e laço de solução de problema ou de análise visual. Um laço de manipulação de dados requer habilidades básicas de coordenação visual (visão) e motora (mãos). Um laço de exploração e navegação requer conhecimentos espaciais acerca objetos de busca. E um laço de análise visual requer conhecimentos acerca o problema de interesse

[1]Colin Ware. *Information Visualization: Perception for Design.* 3a. edição. Cap. 10.

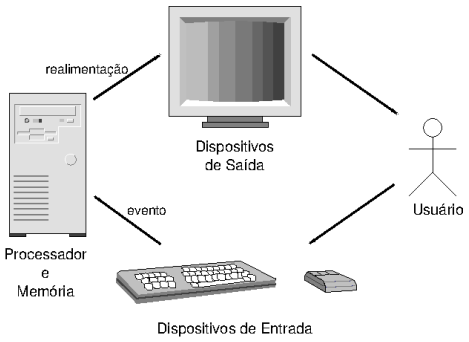


Figura 6.1: Ciclo de Interação.

e as possíveis alternativas de solução. Ware discute ainda as características desejáveis nas interfaces exibidas numa tela bi-dimensional para que estas possam auxiliar a execução efetiva destes laços de interação. Na Seção 6.2 faremos uma síntese destas características. Antes, será apresentada brevemente na Seção 6.1 uma classificação lógica dos periféricos de captura dos eventos gerados por um usuário a fim de entenderem melhor as restrições impostas pelos periféricos (físicos) de interação.

Há dois pontos críticos no ciclo de interação mostrado na Fig. 6.1. Ambos são relacionados com a possível discrepância entre a capacidade de processamento de uma máquina e a capacidade de processamento humano. De um lado, a máquina pode até tomar ações corretas, porém exibir os resultados de forma ilegível ou de forma mais lenta que a expectativa, prejudicando a sua interpretação por parte dos usuários humanos. De outro lado, o usuário pode gerar uma sequência de ações ambíguas ou inconsistentes que não consiga fazer o processador alcançar o estado almejado. Um bom projeto de interface homem-máquina deve atenuar as barreiras, melhorando a usabilidade de um sistema gráfico. Quais considerações tecnológicas devemos levar em conta neste projeto para que ele seja factível? Na seção 6.3 apresentaremos conceitos básicos de um sistema de janelas (X) e detalharemos o seu processamento de eventos, cuja compreensão facilita o entendimento do modelo de programação orientada a eventos.

Ilustraremos na Seção 6.4 a construção de uma simples interface gráfica com uma área (janela) de desenho delegada para OpenGL [2] através de uma biblioteca (básica) de componentes gráficos, desenhada originalmente para o sistema de janelas X GLUT [3]. Finalmente, apresentaremos na Seção 6.5

[2] `http://www.opengl.org/`

[3] `http://www.opengl.org/resources/libraries/glut/`

um sistema de análise visual para diagnóstico de lesões cerebrais displásticas que foi desenvolvido por um grupo do Departamento de Engenharia de Computação e Automação Industrial da FEEC-Unicamp.

6.1 Dispositivos Lógicos de Entrada

Sob o ponto de vista de programação, é interessante que o acesso aos periféricos ocorra de forma mais independente possível das características tecnológicas de cada um. O conceito de dispositivos lógicos foi introduzido para permitir que os programas "vejam" os dispositivos pelas suas funções e não pela sua especificação técnica. De acordo com o padrão gráfico GKS (*Graphical Kernel System*)[4], distinguem-se seis dispositivos de entrada lógicos:

locator: retorna a posição de um ponto da imagem. Exemplos de dispositivos físicos que realizam esta tarefa básica são *mouse* e teclado. Outros dispositivos conhecidos são *joystick*, *trackball* e *tablet*.

stroke: retorna uma sequência de pontos. Tipicamente, esta tarefa pode ser realizada com uso de um *mouse*.

pick: retorna a referência de um objeto contido na imagem exibida pelo dispositivo de saída. O dispositivo físico que tipicamente realiza esta função é o *light-pen*.

valuator: retorna um valor numérico (escalar). Exemplo típico de um *valuator* são os *dials* (potenciômetros).

string: retorna uma sequência de caracteres alfa-numéricos. O dispositivo físico típico que realiza esta tarefa é o teclado.

choice: retorna uma opção dentre um conjunto de alternativas pré-definidas. Exemplo típico de um dispositivo físico que realiza esta tarefe é o *tablet* com funções pré-definidas. É comum, hoje em dia, mapear um conjunto de alternativas às teclas do teclado, tornando-o também um dispositivo lógico de *choice*. Podemos ainda utilizar o *mouse* como um dispositivo lógico de *choice*, se mapearmos cada um dos seus botões a uma das funções pré-definidas ou provermos na interface do aplicativo sub-janelas com as opções pré-definidas.

[4] `https://en.wikipedia.org/wiki/Graphical_Kernel_System`

Um sistema gráfico interativo suporta, usualmente, mais de uma classe de dispositivos lógicos. No mínimo, *locator* (*mouse*) e *string* (teclado).

Considera-se ainda que os dispositivos lógicos podem operar em um dos três modos: **requisição**, **amostragem** e **evento**. No modo de requisição, o aplicativo habilita o dispositivo e fica aguardando a entrada de um dado. No modo de amostragem, o aplicativo amostra o estado do dispositivo periodicamente. E no modo de evento, o dispositivo dispara o processamento de eventos quando há algum dado disponível. Na seção 6.3 detalharemos o processmanto pelo modo de entrada por eventos.

6.2 Laços de Interação

Pela lei de poder de prática (*power law of practice*), sabe-se que o domínio, ou a habilidade, no manuseio de uma ferramenta é adquirido com treinamentos. Algumas tarefas requerem mais tempo de aprendizado (*shallow learning curve*) e outras, menos tempo (*steep learning curve*). No projeto de interfaces que suportam laços de interação, procura-se minimizar este tempo de aprendizado tirando proveito das habilidades natas do nosso corpo. Além disso, Ware chama atenção a alguns aspectos relevantes no desenho de laços de interações:

"*navigating a data space as quickly and transparently as possible. Doing so involves supporting eye–hand coordination, using well-chosen interaction metaphors, and providing rapid and consistent feedback. The word transparent in user interface design is a metaphor for an interface that is so easy to use that it all but disappears from consciousness, but transparency can also come from practice, not just good initial design.*"

"*Simply shortening the amount of time it takes to acquire a piece of information may seem like a small thing, but human visual and verbal working memories are very limited in capacity and the information stored is easily lost; even a few seconds of delay or an increase in the cognitive load can drastically reduce the rate of information uptake by the user. When a user must stop thinking about the task at hand and switch attention to the computer interface itself, the effect can be devastating to the thought process.*"

"*a common bottleneck in cognitive processing comes from the limitations of working memory capacity. This is the reason why it is so essential to enable people to rapidly move attention from one meaningful pattern to another. If we could hold more in our heads, we would not need to shift attention so often. Providing interactive methods that work around the limits of working memory capacity can, in many cases, result in impressive gains*

in efficiency."

"*At the stage of new discoveries in information visualizations, standardization is the enemy of innovation and innovation is the enemy of standardization. It is important that we get the research done before the standards are formed; otherwise, it will be too late. These are exciting times for information visualization, because we are still in the discovery phase ...*"

Faremos a seguir uma síntese das recomendações dadas no livro de Colin Ware.

No nível do laço de manipulação (direta) dos dados, foca-se na ergonometria dos dispositivos físicos, mais especificamente na coordenação dos olhos com as mãos. Há estudos que mostram que o tempo de reação a um evento varia logaritmicamente com relação à quantidade de alternativas apresentadas. Quanto maior é a quantidade de alternativas, maior será o tempo demandado para uma resposta. Este tempo aumenta ainda com o grau de acurácia exigido na resposta. Outra constatação interessante nos estudos é que o tempo de seleção de um item numa tela está relacionado com o índice de dificuldade (*index of difficult*), que por sua vez depende da largura do alvo e da razão da distância entre um *locator* e o alvo. Ou seja, o índice de dificuldade aumenta com o deslocamento que se deve fazer para levar o *locator* ao alvo e diminui com o aumento do tamanho do alvo.

Estendendo o conceito para rastreamento de um caminho exibido numa tela digital através de um *locator*, a velocidade do rastreamento é diretamente proporcional à largura do caminho. Ware comenta ainda que a exibição automática de uma mensagem sucinta sobre os dados quando o dispositivo *locator* passa por eles (*hover queries*) pode aprimorar interações diretas. Outro fator relevante é a consistência entre as ações das mãos sobre um dispositivo lógico e as realimentações visuais na tela. Espera-se, por exemplo, que os objetos movimentam no mesmo sentido em que a mão (manipulando um *locator*) moveu. Ware também discute vantagens e limitações de desenhar uma interface que faz uso de duas mãos. Estudos revelam que se deve usar a mão dominante na execução de tarefas propriamente ditas e deixar para a mão não-dominante o controle de referencial.

No nível intermediário, o de exploração e navegação, ocupa-se com o deslocamento entre os dados até encontrar o alvo. Neste caso, o papel de cognição espacial é fundamental (Fig. 10.3 no livro de Colin Ware[1]). Quando os dados são **físicos**, podemos mapeá-los num espaço tridimensional e reduzir o problema a um problema de busca do rotas e locomoção ao longo destas rotas. Neste caso, uma visualização contínua, a taxa entre 1 a 2 quadros por segundo, da cena 3D em que o alvo se encontra, com adequadas sinalizações (*landmarks*) e referências espaciais, propicia a percepção e

entendimento da posição e da orientação do alvo.

Ware distingue duas classes de referências espaciais e quatro paradigmas de locomoção ao longo das rotas. As duas classes de referenciais são (1) referência egocêntrica, centrada no observador (constituída pela linha de visão, rotação da cabeça frente–trás (*tilt*) e rotação do tronco direita–esquerda (*pan*)), e (2) referência exocêntrica, centrada num ponto espacial fixo. E os quatro paradigmas de locomoção são: (1) o observador se mantém fixo enquanto a cena movimenta (*world-in-hand*), (2) a cena se mantém fixa enquanto o ponto de vista do observador movimenta (*eye-in-hand*), (3) a cena se mantém fixa enquanto o observador movimenta (*walking*), e (4) a cena se mantém fixa enquanto o observador sobrevoa a cena (*flying*).

Quando os dados são **abstratos**, Ware sugere usar os conceitos de foco e contexto no lugar de alvo e cena 3D, respectivamente. Além disso, ao invés de uma única escala espacial, acrescentam-se duas outras escalas para caracterizar melhor os atributos dos dados: escala estrutural e escala temporal. Quando estes dados são mapeados em elementos gráficos, como vimos no Capítulo 5, as mesmas técnicas de interação são aplicáveis com a vantagem das imagens poderem ser deformadas para aumentar a taxa de tinta de dados sem comprometer a sua interpretação física familiar, como ilustram as Figs. 10.10, 10.11, 10.12 e 10.14 no livro de Ware[1].

Finalmente, no nível de análise visual, o objetivo é a tomada de decisão assitida por computador. O componente crucial neste nível de laço é a memória de longa duração onde são armazenados os conhecimentos e a memória de trabalho visual onde são construídos padrões visuais a partir das realimentações visuais às interações de um usuário [5]. Sendo a capacidade de armazenamento da memória de trabalho visual bastante limitada, o *frame-buffer* do computador, onde é armazenada a imagem visualizada, é de fato uma extensão da nossa memória de trabalho visual. Portanto, o computador assiste às nossas tomadas de decisão não só com processamento numérico como também com a extensão da nossa memória. Ware discute em detalhes como o nosso sistema de memórias subisidia a construção de um raciocínio visual, junto com os outros componentes de um **sistema cognitivo visual assistido por computador**:

- pré-processamento visual em canais de características (cor, orientação/forma, textura, e movimento),
- percepção de padrão (leis de Gestalt),
- movimentos de olhos,

[5] Colin Ware. *Information Visualization: Perception for Design.* 3a. edição. Cap. 11.

- laço de escaneamentos sacádicos dos olhos,
- construção de imagens mentais simples,
- simples ações motoras e visuais coordenadas,
- formulação de hipóteses visuais (*visual queries*), e
- algoritmos de mapeamento de dados em elementos gráficos.

6.3 Sistema de Janelas

Um dos desafios tem sido projeto e implementação de uma interface amigável entre um "processador digital" e um "controlador humano", aproveitando ao máximo a tecnologia multimodal disponível. Nesta seção apresentamos o conceito de **sistema de janelas** que facilita a implementação dessa interface. Um sistema de janelas é uma "camada de software" que gerencia o uso compartilhado dos dispositivos de entrada e de saída entre vários aplicativos, de forma similar a um sistema operacional que gerencia os recursos computacionais entre vários processos. Em um sistema de janelas, uma tela do monitor é considerada uma área de trabalho através da qual um conjunto de aplicativos pode interagir com o usuário e cada aplicativo tem uma sub-área própria, como se a tela do monitor estivesse dividida em várias subáreas totalmente independentes (Figura 6.2). Estas sub-áreas, usualmente retangulares, são conhecidas como **janelas** e são organizadas hierarquicamente em árvore, sendo a tela considerada a **janela-raíz**. Todas as janelas são referenciáveis por meio de identificadores (ID).

Existe, em adição ao sistema de janelas, um **gerenciador de janelas** que provê facilidades para usuários personalizar a aparência das janelas e a forma de interações através dos periféricos. Em Microsoft Windows este gerenciador é integrado ao sistema de janelas, enquanto em X Windows, o gerenciador é um componente à parte. Exemplos de gerenciador de janelas para X Windows: Motif Windows Manager, Tab Windows Manager e Window Maker.

As ações dos usuários, como movimento de *mouse*, aperto/soltura de um botão de *mouse*, pressionamento de uma tecla de teclado e focalização de uma janela, são traduzidas em **eventos** e despachados para a janela apropriada para serem tratadas. Em Microsoft Windows tanto o programa aplicativo quanto a interface com usuários estão residentes numa mesma máquina. O X Windows, usado frequentemente em conjunto com o sistema operacional UNIX/Linux, opera segundo o **modelo cliente-servidor**: o

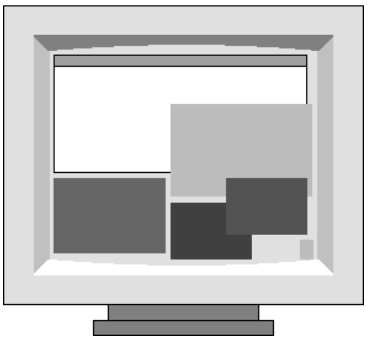


Figura 6.2: Uma área de trabalho com multijanelas.

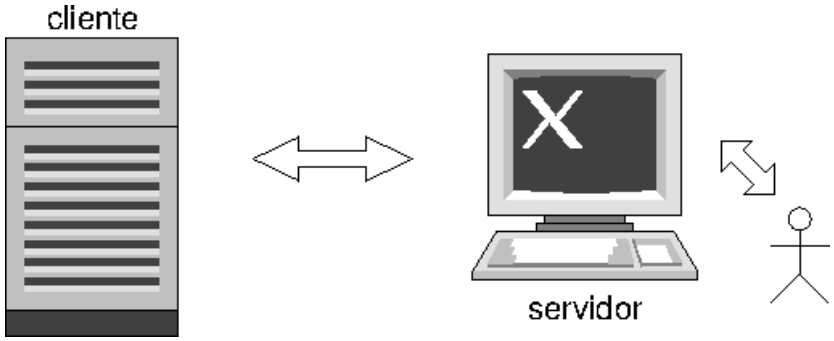


Figura 6.3: Modelo cliente-servidor.

servidor X "serve" vários programas aplicativos, conhecidos como **cliente** X, recebendo as suas requisições de saída gráfica como também enviando para eles os eventos gerados pelo usuário. O servidor e o cliente podem estar instalados em uma mesma máquina ou em máquinas distintas. Este modelo de comunicação permite o uso de janelas e o controle de aplicativos de modo transparente através da rede (Figura 6.3).

No restante desta seção vamos detalhar o processamento de eventos.

6.3.1 Fila de Eventos

Como se consegue "sincronizar" as ações não-programáveis de um usuário com uma sequência de instruções pré-estabelecidas em um programa aplicativo? Uma solução popular é por meio de uma **fila (principal) de eventos (de entrada)**, como em X Windows. Após traduzir as ações em eventos, estes são inseridos em uma estrutura. O **tratador de eventos** os remove da

estrutura de acordo com o princípio FIFO (*first in, first out*) e os converte em **eventos de interação** antes de despachá-los para serem tratados por um bloco de instruções. O tratador de eventos pode tanto concatenar uma sequência de eventos de entrada em um único evento de interação, como CTRL+ALT+Del e duplos cliques, quanto mascarar eventos irrelevantes. Exemplos de algumas classes de eventos processáveis em todos os sistemas de janela: eventos de dispositivo de apontamento como *mouse*, eventos de teclado e eventos de janela, como criar, destruir e focalizar uma janela.

6.3.2 Despachador de Eventos

Sendo a tela do monitor fragmentada em diversas janelas, muitas delas compartilhando o mesmo conjunto de *pixels*, é necessário elaborar uma estratégia de associação entre eventos de interação com trechos de código de processamento para obter resposta apropriada. Quando a posição do *cursor* em relação a uma janela não é ambígua, esta associação é simples. Em alguns sistemas a informação posicional do apontador é utilizada para despachar também os eventos do teclado: eles são encaminhados para a janela que contém o *cursor*. O problema se complica quando existe mais de uma janela sob o *cursor*. Neste caso, qual deve ser o critério de escolha da janela? Uma solução é a introdução do conceito de **foco**: somente uma janela pode estar em foco em cada instante e todos os eventos gerados num dado momento são despachados para a janela em foco.

6.3.3 Tratador de Eventos

Um ciclo de interação envolve o disparo de um evento de interação, o despacho deste evento à janela correta e a execução de um trecho de código condizente com a ação esperada (Figura 6.1). A sequência de instruções a serem executadas deve ser, portanto, dependente do contexto de aplicação. Como se integram estas instruções ao restante dos códigos de processamento de eventos? Uma solução é pelo mecanismo callback que permite passar como argumento o endereço do tratador de eventos, provido pela aplicação, para o sistema de janelas. Assim, quando um evento é disparado, o sistema de janelas automaticamente chama o seu correspondente tratador para executar a tarefa pré-programada.

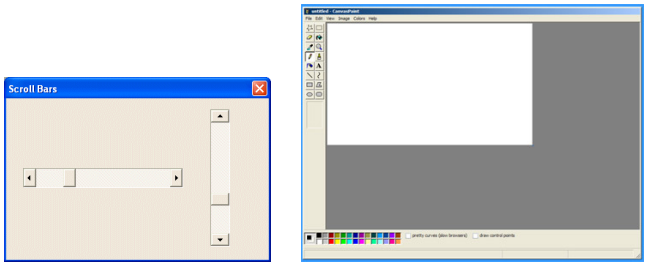
6.4 Bibilioteca de Componentes de Interações

Entendendo a arquitetura de um sistema de janelas, fica mais fácil entender a forma como um programa orientado a eventos é estruturado. Ele possui sempre um programa inicial contendo instruções de inicialização e um **laço principal de eventos** cuja tarefa é simplesmente retirar os eventos da fila e despachá-los. As funções de inicialização são as instruções comuns a todos os aplicativos, como mascaramento de eventos irrelevantes, registro de função callback[6] e criação de uma janela principal. O comportamento real de um aplicativo é, de fato, definido pelos tratadores de evento registrados com callback.

Para reduzir o tempo de desenvolvimento de uma interface gráfica com usuário (GUI – *graphical user interface*), são disponíveis bibliotecas de componentes de interface gráfica, ou widgets em inglês, em cima de um sistema de janelas. Um widget é um componente capaz de realizar uma ou mais tarefas de interação com um tipo específico de dados. Tipicamente, um *widget* é desenhado com base no padrão de projeto de *software* **MVC**, constituído de dois componentes distintos, **M**odelo de dados e **V**isualização ou aparência gráfica, e um componente de **C**ontrole ou modo de interação. Uma barra de rolagem é, por exemplo, um widget que tem como modelo, um intervalo de valores numéricos dependents do aplicativo, como aparência um deslizador, e como controle, a possibilidade de um usuário "deslocar" o botão do deslizador que pode ser traduzido, por exemplo, como "rolar" uma área de dimensões maiores do que a da janela onde ela é exibida (Figura 6.4.(a)). Em especial, o widget de tela (Canvas widget) é um componente que tem aparência de uma tela e provê facilidades gráficas 2D e/ou 3D para criar e manipular livremente os modelos da dados do aplicativo (Figura 6.4.(a)). Entre as bibliotecas de widgets amplamente utilizadas temos GTK+, wxWidgets, Qt e Swing da linguagem de programação Java.

Utilizaremos a biblioteca de widgets GLUT (*OpenGL Utility Toolkit*)[3] para implementar simples interfaces de aplicativos gráficos 3D. Além de ser independente do sistema de janelas, GLUT provê um widget de tela capaz de se comunicar a interface de programação de aplicativos OpenGL[2] e utiliza o modelo de callback para registrar os tratadores de eventos, como ilustra o seguinte esboço de código extraído. O arquivo glut.h deve ser sempre incluído para definir os macros e constantes utilizados. Com uso das funções glutInitWindowSize(), glutInitWindowPosition() e glutCreateWindow(), foi definida uma janela "hello" de tamanho 250×250 em $(100, 100)$ que se comunica com

[6] A denominação varia entre sistemas de janelas.



(a) (b)

Figura 6.4: *Widgets*: (a) barra de rolagem e (b) tela.

API OpenGL e registrado o evento de criar esta janela. Somente o evento de janela não é mascarado e o seu tratador display() é registrado através da função glutDisplayFunc(). O tratador simplesmente desenharia os elementos gráficos com os comandos de OpenGL dentro da rotina display() quando a janela sofre alguma alteração. O laço principal de eventos glutMainLoop() fica "aguardando" tais eventos. Ressaltamos que a janela definida só aparece na tela quando entra no laço principal de eventos e o evento de "criar a janela" é processado.

```
#include <GL/glut.h>

void display(void)
{
/* Triggering OpenGL rendering */
}

void init (void)
{
/* Configure OpenGL pipeline and load data buffers */
}

/*
 * Declare initial window size, position, and display mode
 * (single buffer and RGBA).  Open window with "hello"
 * in its title bar.  Call initialization routines.
 * Register callback function to display graphics.
 * Enter main loop and process events.
 */
```

```
int main(int argc, char** argv)
{
   glutInit(&argc, argv);
   glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);
   glutInitWindowSize (250, 250);
   glutInitWindowPosition (100, 100);
   glutCreateWindow ("hello");
   init ();
   glutDisplayFunc(display);
   glutMainLoop();
   return 0;   /* ANSI C requires main to return int. */
}
```

Vale, porém, ressaltar que todos estes trabalhos de implementação com uso de bibliotecas de componentes de interação só se aplicam quando se desenvolve um sistema *from scratch*. Hoje em dia existem ferramentas de visualização, como ParaView [7] e VTK [8], que permitem uma rápida prototipação do seu projeto sem se preocupar com a infra-estrutura das interações.

6.5 Projeto de um Sistema de Análise Visual

Até aqui apresentamos os principais ingredientes de um sistema de análise visual. Vamos mostrar agora como aplicá-los para concretizar um aplicativo que envolve análise visual.

Colin Ware apresenta no seu livro[5] esboço do projeto de 9 aplicativos que envolvem análise visual (*visual thinking algorithms*). Compartilho aqui experiências que tivemos com o desenvolvimento do aplicativo VMTK [9], em termos dos 5 itens destacados pelo Ware:

Operações cognitivas: localizar tecidos cerebrais anormais em pacientes epilépticos [10].

Informações visualizadas: volumes multimodais co-registrados são refatiados multiplanar- ou curvilinearmente. Quatro vistas do volume são renderizadas: 3D, vista axial, vista coronal e vista sagital. Para que o sistema seja responsivo, múltiplos dados não visualizados, como *depth*

[7] `https://www.paraview.org/`

[8] `https://vtk.org/`

[9] `http://www.dca.fee.unicamp.br/projects/mtk/vmtk-neuro/index.html`.

[10] C.L.Yasuda e F. Cendes. *Neuroimaging for the prediction of response to medical and surgical treatment in epilepsy.* `https://www.ncbi.nlm.nih.gov/pubmed/23480740`

map [11], são renderizados de forma transparente, e os volumes multimodais são pré-carregados em unidades de processamento gráfico (GPU) para que a imagem co-registrada seja renderizada integralmente na GPU [12]. Foi usado originalmente a biblioteca WxWidget [13], depois migramos para Qt pelas funções disponíveis [14], na construção de interfaces com usuários.

Ações Epistêmicas: *mouse* é usado para explorar o volume no modo *world-in-hand* e refatiar o volume em diferentes angulações. Sequência de ações permissíveis é modelada por um diagrma de estados para evitar ações indevidas.

Externalização: configurar o limiar para remover o ruído do fundo de cada volume, alinhar inicialmente os volumes multimodais, definir a região de refatiamento curvilíneo.

Computação: co-registrar volumes multimodais [15], renderizar volumes multimodais, reformatar multiplanar- ou curvilinearmente [16].

6.6 Exercícios

1. A quais classes de dispositivos lógicos um *mouse* e um teclado pertencem? Justifique.

2. Cite dois aplicativos gráficos interativos em que o laço de interação é o de manipulação de dados. Justifique.

3. No texto está escrito que uma "navegação numa cena 3D em que o alvo se encontra, com adequadas sinalizações, propicia a percepção e o entendimento da posição e da orientação do alvo". O que seriam as sinalizações adequadas? Justifique.

---

[11] S.-T, Wu e colegas. *Snapping a Cursor on Volume Data.* `https://dl.acm.org/citation.cfm?id=2121375`

[12] S.T. Wu e colegas. *Toward a Multimodal Diagnostic Exploratory Visualization of Focal Cortical Dysplasia].* `https://ieeexplore.ieee.org/document/8370206/`

[13] `https://www.wxwidgets.org/`

[14] `https://www.qt.io/`

[15] S.-T. Wu e colegas. *Pre-alignment for Co-registration in Native Space.* `https://ieeexplore.ieee.org/document/6915288`

[16] S.-T. Wu e colegas. *Interactive patient-cuetomized curvilinear reformatting for improving neurosurgical planning.* `https://www.ncbi.nlm.nih.gov/pubmed/30343394`

4. Qual é o modo de "locomoção" adotado no ParaView para explorar um objeto visualizado? Justifique.

5. Engloba o laço de interação de análise visual os laços de interação de manipulação, exploração e navegação de dados? Justifique.

6. Escolha 1 dos 9 projetos descritos pelo Colin Ware[5] e projete uma forma de implementação.

7. Descreva sucintamente o seu projeto final do curso no molde como o Colin Ware apresentou os 9 projetos no seu livro. Destrinche a sua proposta para cada item destacado na Seção 6.5.