Anexo A

Documentação javascript

- Window Manager
- Node Manager
- Window Engine

Universidade de Aveiro
Licenciatura em Tecnologias e Sistemas de Informação
Projecto de Licenciatura
2010-2011
Francisco Alexandre de Gouveia

Índice

Índice de ilustrações

Descrição dos componentes

Os componentes de gestão de janelas e gestão de nós são independentes entre si, sendo possível reutiliza-los noutros contextos e em separado. O componente *engine* faz uso de ambos os componentes para criação e ligação destas de acordo com o necessário.

Para que o componente *windowManager* funcione, é necessário que exista um elemento *div* que servirá de área de trabalho para janelas. Em relação ao componente *nodeManager* funcione é necessário que exista um elemento *canvas* para que este possa desenhar ligações entre nós.

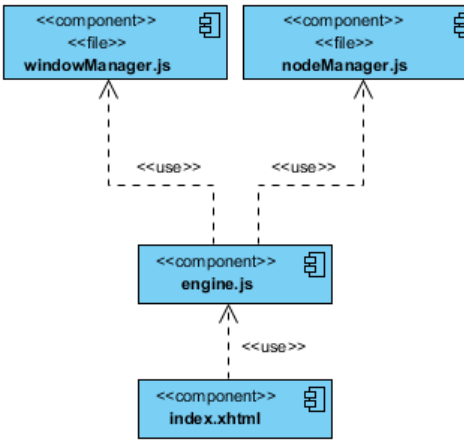

Ilustração 1 - Dependência de componentes

Componente de gestão de janelas

A variável global deste componente é *windowManager*, e tem os seguintes métodos:

init( ) : void

- Método de inicialização do componente. Aqui são definidos:
  - O elemento *div* que contém os nós;
  - A diferença entre a posição do elemento *div* e a coordenada [x=0, y=0], usada para mover os nós (é necessário porque o cálculo da posição dos elementos é feito de forma relativa ao elemento onde está contido);
  - As acções a serem executadas ao mover o rato no elemento *div* "*windowContainer*", usando eventos. Neste caso, estas acções são a obtenção das coordenadas da posição do rato em relação ao elemento *body* e mover o nó activo com o movimento do rato (ver moveWindow() : void). As formas de obter a posição do ponteiro do rato e realizar o movimento da janela foram baseadas no algoritmo apresentado por Mark Kahn [1].

setOnMove(fn) : void

- Argumentos:
  - fn : function - Função a ser executada ao mover um nó.
- Método para definir uma função a ser executada quando um nó é movido.

setOnClose(fn) : void

- Argumentos:
  - fn : function - Função a ser executada ao fechar um nó.
- Método para definir uma função a ser executada quando um nó é fechado.

setOnFocus(fn) : void

- Argumentos:
  - fn : function - Função a ser executada ao seleccionar um nó.
  - Método para definir uma função a ser executada quando um nó é seleccionado.

setOnUnfocus(fn) : void

- Argumentos:
  - fn : function - Função a ser executada ao tirar a selecção de um nó.
- Método para definir uma função a ser executada quando um nó deixa de estar seleccionado.

createWindow(windowName, closeButton) : window

- Argumentos:
  - windowName : string - Texto a apresentar na barra superior do nó.
  - closeButton : boolean - Definir se nó contém botão para fechar.
- Retorno:
  - window : element - Elemento div que representa o nó (ver Elemento nó/janela).
- Método para criação de um novo nó. Aqui é gerado o código html destinado a ser colocado dentro do *windowContainer*, assim como são definidos os métodos deste elemento (ver Elemento nó/janela). São definidos eventos para quando o rato é pressionado e largado, usado para definir qual janela deve acompanhar o movimento do rato e para colocar a janela no topo das outras todas (focusWindow(window) : void). Dentro da janela é criado um elemento div com o fim de receber o conteúdo referente a esta.

putCloseButton(window) : void

- Argumentos:
  - window : element - Nó onde será colocado o botão para fechar nó.
- Método usado para colocar o botão de fechar nó dentro do elemento que representa o nó. É definido o evento que ao carregar chama o método closeWindow (ver closeWindow(window) : void).

dragWindow(window) : void

- Argumentos:
  - window : element - Nó a ser movido.
- Método usado para definir um nó a ser arrastado quando o rato se move. É calculada a diferença entre a posição do rato e a janela, para que o movimento esteja de acordo com a posição inicial do rato em relação à janela quando o movimento começa.

dropWindow( ) : void

- Desactiva o movimento da janela activa.

moveWindow() : void

- Método usado pelo evento de movimento do rato, usado para mover a janela activa de acordo com a posição do rato, usando as coordenadas calculadas anteriormente (ver dragWindow(window) : void). É usado um semáforo para não haver sobreposição de chamamento do método e um atraso de 10 milissegundos no fim de cada movimento da janela, para não haver processamento exagerado. São chamados os métodos definidos no setOnMove da janela e no setOnMove do windowManager, caso tenham sido definidos anteriormente (ver setOnMove(fn) : void).
- A ordem de execução deste método é:
  - onMove da janela (todos os métodos definidos no setOnMove da janela, se existirem);

    - onMove do windowManager (o método definido no setOnMove do windowManager, se existir);
    - Operações do moveWindow.
- Isto significa que caso algum dos métodos definidos no setOnMove retornem o valor *false*, os passos seguintes são ignorados, permitindo assim o cancelamento das operações de movimento.

focusWindow(window) : void

- Argumentos:
    - window : element - Janela a ser colocada em primeiro plano.
- Método usado para colocar uma janela no topo de todas as outras com recurso a estilos, através da propriedade z-index, que posiciona as janelas no eixo Z, ou seja, a janela que tiver esta propiedade com o valor maior, fica em primeiro plano, como está representado na Ilustração 2. São chamados os métodos definidos no setOnFocus da janela e no setOnFocus do windowManager, caso tenham sido definidos anteriormente (ver setOnFocus(fn) : void), além dos métodos definidos no setOnUnfocus na janela que estava anteriormente seleccionada.

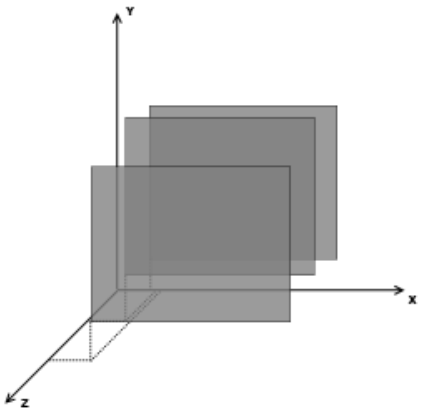

Ilustração 2 - Z-index

- A ordem de execução deste método é:
    - onFocus da janela (todos os métodos definidos no setOnFocus da janela, se existirem);
    - onFocus do windowManager (o método definido no setOnFocus do windowManager, se existir);
    - Operações do focusWindow, incluindo o método onFocus da janela anteriormente seleccionada.
- Isto significa que caso algum dos métodos definidos no setOnFocus retornem o valor *false*, os passos seguintes são ignorados, permitindo assim o cancelamento das operações de selecção.

closeWindow(window) : void

- Argumentos:
    - window : element - Janela a ser fechada.
- Método usado ao carregar no botão de fechar na janela. São chamados os métodos definidos no setOnClose da janela e no setOnClose do windowManager, caso tenham sido definidos anteriormente (ver setOnClose(fn) : void) e é removido no fim o elemento do *WindowContainer*.
- A ordem de execução deste método é:
    - onClose da janela (todos os métodos definidos no setOnClose da janela, se existirem);
    - onClose do windowManager (o método definido no setOnClose do windowManager, se existir);
    - Operações do closeWindow.
- Isto significa que caso algum dos métodos definidos no setOnClose retornem o valor *false*, os passos seguintes são ignorados, permitindo assim o cancelamento das operações de fecho da janela.

onClose( ) : boolean

- Retorno:
  - boolean - Caso se pretenda o cancelamento do fecho da janela, a função definida no *setOnClose* deve retornar o valor *false*.
- Executa o método definido no *setOnClose*.

onFocus( ) : boolean

- Retorno:
  - boolean - Caso se pretenda o cancelamento da selecção da janela, a função definida no *setOnFocus* deve retornar o valor *false*.
- Executa o método definido no *setOnFocus*.

onMove( ) : boolean

- Retorno:
  - boolean - Caso se pretenda o cancelamento do movimento da janela, a função definida no *setOnMove* deve retornar o valor *false*.
- Executa o método definido no *setOnMove*.

onUnfocus( ) : void

- Executa o método definido no *setOnUnfocus*.

Elemento nó/janela

Juntamente com todas as propriedades e métodos do elemento nativo do javascript (HTTPElement), os seguintes métodos foram acrescentados na criação do nó/janela. É de relembrar que este elemento corresponde a um elemento *div*.

centerPosition( ) : {x, y}

- Retorno:
  - {x : number, y : number} - Valores referentes à posição do centro da janela.
- Obtém o valor do centro da janela através do cálculo:
  - $posiçãoJanela + (\frac{taman\ hoJanela}{2})$

toString ( ) : string

- Retorno:
  - string - Título da janela.

setContent(content) : void

- Argumentos:
  - content : string - Texto a ser inserido na janela.
- Usado para inserir texto no elemento *content* da janela.

addElement(element) : void

- Argumentos:
  - element : element - Elemento a ser inserido na janela.
- Usado para inserir elementos no elemento content da janela.

setName(name) : void

- Argumentos:

    - name : string - Novo nome a atribuir à janela.
- Método para renomear a janela.

getName( ) : string

- Retorno:
    - string - Nome da janela.
- Método para obter o nome da janela.

setX(x) : void

- Argumentos:
    - x : number - Posição da janela no eixo do X.
- Método para definir a posição da janela no eixo do X.

getX( ) : number

- Retorno:
    - number - Posição da janela no eixo do X.

setY(y) : void

- Argumentos:
    - y : number - Definir posição da janela no eixo do Y.
- Método para definir a posição da janela no eixo do Y.

getY( ) : number

- Retorno:
    - number - Posição da janela no eixo do Y.

setWidth(width) : void

- Argumentos:
    - width : number - Largura da janela.
- Método para definir a largura da janela.

getWidth( ) : number

- Retorno:
    - number - Largura da janela.

setHeight(height) : void

- Argumentos:
    - height : number - Altura da janela.
- Método para definir a altura da janela.

getHeight( ) : number

- Retorno:
    - number - Altura da janela.

setContentWidth(width) : void

- Argumentos:
    - width : number - Largura do conteúdo da janela.
- Método para definir a largura da janela, segundo a largura do conteúdo (adiciona margens entre o conteúdo e a janela).

getContentWidth( ) : number

- Retorno:
  - number - Largura do conteúdo da janela.

setContentHeight(height) : void

- Argumentos:
  - height : number - Altura do conteúdo da janela.
- Método para definir a altura da janela, segundo a altura do conteúdo (adiciona margens entre o conteúdo e a janela).

getContentHeight( ) : number

- Retorno:
  - number - Altura do conteúdo da janela.

setTopMost(zIndex) : void

- Argumentos:
  - zIndex : number - Posição no eixo do Z (quanto maior, mais à frente fica).
- Método para definir manualmente a posição da janela no eixo do Z. Desta forma, quando a janela perde a selecção não é colocada por trás das outras janelas.

isTopMost ( ) : boolean

- Retorno:
  - boolean - Valor *true* se está definida como *top most*.

removeTopMost ( ) : void

- Coloca a janela ao mesmo nível das outras que não estejam em *top most*.

setOnMove(fn) : void

- Argumentos:
  - fn : function - Função a ser executada ao mover um nó.
- Método para adicionar uma função a ser executada quando um nó é movido.

setOnClose(fn) : void

- Argumentos:
  - fn : function - Função a ser executada ao fechar um nó.
- Método para adicionar uma função a ser executada quando um nó é fechado.

setOnFocus(fn) : void

- Argumentos:
  - fn : function - Função a ser executada ao seleccionar um nó.
- Método para adicionar uma função a ser executada quando um nó é seleccionado.

setOnUnfocus(fn) : void

- Argumentos:
  - fn : function - Função a ser executada ao tirar a selecção de um nó.
- Método para adicionar uma função a ser executada quando um nó deixa de estar seleccionado.

onClose( ) : boolean

- Retorno:
  - boolean - Caso se pretenda o cancelamento do fecho da janela, pelo menos uma das funções definidas no *setOnClose* deve retornar o valor *false*.
- Executa os métodos definidos no *setOnClose*.

onFocus( ) : boolean

- Retorno:
  - boolean - Caso se pretenda o cancelamento da selecção da janela, pelo menos uma das funções definidas no *setOnFocus* deve retornar o valor *false*.
- Executa os métodos definidos no *setOnFocus*.

onMove( ) : boolean

- Retorno:
  - boolean - Caso se pretenda o cancelamento do movimento da janela, pelo menos uma das funções definidas no *setOnMove* deve retornar o valor *false*.
- Executa os métodos definidos no *setOnMove*.

onUnfocus( ) : void

- Executa os métodos definidos no *setOnUnfocus*.

Componente de gestão e ligação de nós

Para ligação entre janelas, foi criado um componente que desenha linhas entre dois pontos/nós. Para isso, este módulo guarda todas as ligações entre dois nós e desenha-os num elemento *canvas* do HTML5. A variável global deste componente é *nodeManager* e tem os métodos descritos nos subcapítulos seguintes.

O elemento *canvas* é colocado dentro do *windowContainer* e é posicionado por trás de todas as janelas, ficando sempre no fundo. Para se colocar um elemento canvas basta usar a seguinte *tag* HTML[2]:

```
<canvas id='canvasId'>
        <p>Mensagem de erro (e.g.: Este browser não suporta canvas)</p>
</canvas>
```

Ilustração 3 - Elemento canvas (HTML5)

Caso o browser não reconheça a *tag*, trata-a como um elemento normal sem formatação, mostrando os elementos nele contidos normalmente. Caso contrário, os elementos nele contido são ignorados e é criado um espaço de desenho no browser.

Para se começar a desenhar, é necessário ter um contexto. Ao criar um novo contexto, o canvas é apagado. Para se criar um contexto para desenho bidimensional, usa-se o método *elementoCanvas.getContext('2d')*.

Para desenhar uma linha é necessário iniciar um percurso com o método *contexto.beginPath()*. A primeira instrução deverá ser a de posicionar o ponto nas coordenadas onde a linha começa, com o método *contexto.moveTo(x, y)*. Para desenhar a linha, usa-se *contexto.lineTo(x, y)* com as coordenadas onde a linha acaba. Pode-se usar várias vezes o método *lineTo* para desenhar várias linhas contínuas. Para pintar as linhas, usa-se *contexto.stroke()*.

Para terminar, fecha-se o percurso com *contexto.closePath()*.

Na Ilustração 4 está implementado o código completo para desenhar linhas de ligações entre nós.

```
var context = elementoCanvas.getContext('2d');
context.strokeStyle = '#000000';
context.lineWidth=1;
for(n in nodeManager.nodes){
        var node = nodeManager.nodes[n];
        context.beginPath();

        context.moveTo(node.a.x, node.a.y);
  //node a
        context.lineTo(node.b.x, node.b.y);
  //node b
        context.stroke();

        context.closePath();
}
```

Ilustração 4 - Desenho em canvas

init(canvas) : void

- Argumentos:
  - canvas : HTMLElement - Elemento onde serão desenhadas linhas entre nós.
- Método de inicialização do componente. Aqui são definidos:
  - O elemento *canvas* onde serão desenhadas as ligações entre os nós;
  - Definição de métodos do para os objectos *Connection* (ver Objecto de ligação entre dois nós).

connectNodes(a, b) : void

- Argumentos:
  - a : object - Nó parente;
  - b : object - Nó descendente de *a.*
- Caso a ligação ainda não exista, cria uma nova ligação do nó *a* ao nó *b*.

disconnectNodes(a, b) : void

- Argumentos:
  - a : object - Nó;
  - b : object - Nó.
- Caso a ligação exista, remove-a.

disconnectNode (node) : void

- Argumentos:
  - node : object - Nó.
- Caso o nó *node* exista, este é removido juntamente com os seus descendentes

exist (a, b) : boolean

- Argumentos:
  - a : object - Nó;
  - b : object - Nó.
- Retorno:

    - boolean - Retorna valor *true* caso a ligação exista.
- Verifica se a ligação entre estes nós existe.

getChildNodes (node) : array

- Argumentos:
    - node : object - Nó.
- Retorno:
    - array - Lista de descendentes directos de *node*.

getParentNodes (node) : array

- Argumentos:
    - node : object - Nó.
- Retorno:
    - array - Lista de parentes de *node*.

drawOnCanvas ( ) : void

- Desenha no elemento canvas linhas entre as posições centrais dos nós. Sempre que este método é chamado, o canvas é limpo para redesenhar.

Objecto de ligação entre dois nós

Este elemento é constituído por dois objectos, correspondente a dois nós distintos. Aqui são guardadas as ligações entre esses nós, e é assumido que o primeiro nó é parente do segundo nó (no contexto deste projecto não existem ligações entre nós vizinhos).

Connection(a, b)

- Argumentos:
    - a : object - Nó parente;
    - b : object - Nó descendente do nó *a*.
- Constructor para o objecto.

toString ( ) : string

- Retorno:
    - string - Coordenadas do nó *a* e do nó *b*.

is (a, b) : boolean

- Argumentos:
    - a : object - Nó;
    - b : object - Nó.
- Retorno:
    - boolean - Retorna o valor *true* caso a ligação *Connection.a* for *a* e *Connection.b* for *b*, ou Connection.a for b e Connection.b for *a*.
- Usado para saber se já existe a ligação entre esses nós.

has(node) : boolean

- Argumentos:
    - node : object - Nó.
- Usado para saber se o nó pertence à ligação.

Bibliografia

Mark Kahn. How to drag and drop in javascript. [Online].
1] http://www.webreference.com/programming/javascript/mk/column2/

Mihai Sucan. (2009, Janeiro) Dev.Opera HTML5 canvas - the basics. [Online].
2] http://dev.opera.com/articles/view/html-5-canvas-the-basics/