Criação de elementos interativos 2D para API's gráficas utilizando CEGUI

Daniel. M. Tortelli Jorge. L. Salvi Maikon. C. Santos Jacques. D. Brancher

Universidade Regional Integrada do Alto Uruguai e das Missões – Campus de Erechim, Brasil

Figura 1: Imagem utilizando elementos gráficos interativos da CEGUI.

Abstract

The objective of this paper is present the interactive creation process stages of the elements 2D (widgets) for graphical API's 3D, as OpenGL, Ogre 3D. The given emphasis is referring to the interface elements, indispensable for the user-system interaction created in a computational game. The functioning of CEGUI library for creation of GUI is shown (Graphical User Interface).

Keywords: *widgets*, CEGUI, Ogre 3D, *Graphical User Interface*, interatividade

Authors' contact:
{daniel.beka, maikoncs}@gmail.com
{j_brancher, jlsalvi}@hotmail.com

1. Introdução

A interface com o usuário é uma parte importante para qualquer tipo de aplicação digital, principalmente para jogos. Não há nenhum jogo de computador que não possua algum tipo de interface. Seja ela apenas textual, o que é comumente encontrado em MUDs (*Multi User Dungeons*) ou complexa que permite-nos construir unidades (*RTS*), escolher uma ação para o personagem principal (*Adventures*), ou dar-nos importantes informações como a altitude atual de uma aeronave (*Fligh-sims*).

Elementos gráficos interativos, que são parte do sistema de uma GUI, são os responsáveis pela interação entre o usuário e a aplicação. Eles modificam o estado da mesma, através de eventos que são gerados em resposta as ações realizadas pelo usuário nesses elementos.

Em aplicações gráficas interativas, como por exemplo jogos de computador, é necessário que os elementos interativos sejam originais e o mais intuitivos possível para o usuário, o que influencia diretamente na qualidade da jogabilidade.

Durante o processo de criação de um jogo, uma das etapas necessárias é o desenvolvimento de um sistema de GUI. API's gráficas como OpenGL e Ogre3D não dispõe desse recurso nativo, necessitando que ele seja criado e incorporado como parte do motor do jogo.

Entretanto, o desenvolvimento de um sistema de uma GUI consome um tempo considerável no projeto de um jogo. Uma solução alternativa para quem faz uso dessas API's para criação de aplicações gráficas interativas é utilizar sistemas prontos de GUI, como o Sistema de GUI denominado CEGUI.

O objetivo deste artigo é apresentar as etapas de criação de um sistema de GUI utilizando a biblioteca CEGUI.

2. Bibliotecas para elementos interativos em jogos

As bibliotecas são compostas de um conjunto de instruções que são utilizadas pela aplicação para a realização de determinadas tarefas, como por exemplo, criação de janelas, gerenciamento de áudio, redes, Inteligência Artificial, entre outras.

Existem algumas bibliotecas disponíveis gratuitamente que oferecem as ferramentas necessárias para a criação de um sistema de GUI para aplicações gerais. No caso específico de jogos, estas são poucas e na maior parte dos casos, encontram-se ainda em fase de desenvolvimento.

Foram relacionadas algumas dessas bibliotecas para identificar qual delas seria adotada no desenvolvimento da interface. A avaliação foi feita com: CEGUI, GTK+ e Guichan GUI. Neste processo, foi levado em conta algumas características das mesmas, entre elas: Linguagem de programação, plataforma, código aberto (*open-source*), documentação e ajuda ao desenvolvedor.

A Tabela 1 (GAME PROGRAMMING WIKI, 2006) ilustra as diferenças e semelhanças das bibliotecas avaliadas. Assim, foi possível fazer a escolha com base nas vantagens que uma biblioteca de interface ofereceu em maior quantidade, em relação às outras.

BIBLIOTECAS PARA CRIAÇÃO DE SISTEMAS DE GUI			
	CEGUI	**GTK+**	**Guichan GUI**
Linguagem	C++	C++, Pearl, Phyton	C++
Plataforma	Windows, Linux, Maços	Linux, Windows	Windows
Licença	LGPL, MIT	LGPL	BSD
Código Aberto (Open-Source)	Sim	Sim	Sim
Documentação	API + Tutoriais	API + Tutoriais	API
Ajuda ao usuário/desenvolvedor	Fórum	Lista de Discussão	Lista de Discussão
Última Atualização	13/08/2006	05/09/2006	29/07/2006
FAQ	Sim	Sim	Sim
Possibilidade de modificar a aparência dos widgets	Sim	Sim	Sim
Aplicações que usam a biblioteca	Ogre 3D (OGRE 2006)	GUIMP (GUIMP 2006)	The Mana World (MW 2006)

Tabela 1: Características de algumas biblitecas para criação de um sistema de GUI

Como foi apresentada, a CEGUI apresentou vantagens em relação às outras bibliotecas no que se refere aos critérios plataforma, documentação e ajudo ao desenvolvedor. Por isso, a mesma foi escolhida para o desenvolvimento da interface do jogo.

3. CEGUI (*Crazy Eddie's GUI System*)

A CEGUI é uma biblioteca *open-source* que provê elementos gráficos que permitem a interação entre o usuário e a aplicação (jogo de computador), provendo as ferramentas necessárias para a criação um sistema de GUI (*Graphical User Interface*).

Esses elementos gráficos interativos, também chamados de *widgets*, são muito utilizados em sistemas operacionais gráficos como, por exemplo, o Windows. Os mais comuns são: *scroll bars, radio buttons, check box, input box, command buttom, sliders, list box, combo box, mouse arrow*, entre outros.

A CEGUI, como qualquer outro sistema de GUI, trabalha orientada a eventos. Os eventos são acionados por certas ações, tais como: pressionamento de teclas, cliques no mouse, deslizamento de uma barra de rolagem, entre outros. Os eventos são registrados em funções específicas. Por exemplo, quando o jogador realiza uma interação com algum *widget*, um evento específico é acionado e a função correspondente realiza a ação apropriada.

A CEGUI é flexível na escolha do renderer. Atualmente, a CEGUI pode ser renderizada em DirectX 8, DirectX 9 e OpenGL.

Segundo CEGUI [2006], "a biblioteca é voltada principalmente para desenvolvedores de jogos de computador que desejam acrescentar elementos interativos em seus jogos, sem que tenham que se preocupar com a criação de um sistema de GUI."

Como todo projeto *open-source* há uma licença. Pode-se usar a CEGUI em projetos comerciais ou não-comerciais exigindo que a biblioteca seja ligada dinamicamente ao projeto que está sendo desenvolvido.

4. Como a CEGUI funciona

O processo de criação dos *widgets* compreende etapas distintas, que são interligadas para formar o esquema completo do sistema de GUI. São etapas do processo:

- Criação da aparência dos elementos gráficos interativos;
- Criação do arquivo *Imageset*;
- Criação do arquivo *LooknFeel*;
- Criação do arquivo *Font*;
- Criação do arquivo *Layout*;
- Criação do arquivo *Scheme*;

4.1. Criando a aparência dos *widgets*

A aparência dos elementos gráficos é definida em um arquivo de imagem nos formatos PNG ou TGA (figura 2). Esses formatos são utilizados por suportarem o canal *alpha*, responsável pela transparência. Alguns elementos como *input Box, list Box, window background* possuem uma única aparência. Outros, como o caso dos *command buttons, radio buttons* e *check box*, necessitam que seja definida a aparência de cada um de seus 3 possíveis estados: desativado, selecionado, ativado.

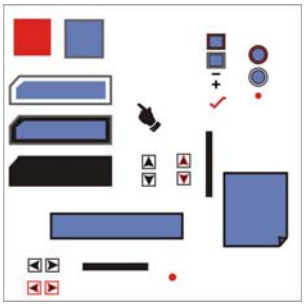

Figura 2: Imagem contendo a aparência dos elementos gráficos interativos com seus possíveis estados.

4.2. Mapeamento dos *widgets* e criação do arquivo *Imageset*

Nessa etapa é feito o mapeamento do arquivo de imagem (figura2), gerando um arquivo XML que descreve um conjunto de *widgets* que serão criados, juntamente com suas posições, tamanhos e outros atributos.

O algoritmo 1 representa a definição de um *Check Box* com seus três estados (desativado, selecionado, ativado), suas posições mapeadas do arquivo de imagem e seus respectivos nomes:

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <Imageset Name="Widgets" Imagefile="widgets.tga"
  NativeHorzRes="800" NativeVertRes="600"
  AutoScaled="true" >
3 <Image Name="CheckboxHover"   XPos="348" YPos="50"
  Width="32" Height="28" />
4 <Image Name="CheckboxMark"     XPos="352" YPos="157"
  Width="30" Height="22" XOffset="9" YOffset="-7"/>
5 <Image Name="CheckboxNormal" XPos="348" YPos="82"
  Width="32" Height="28" />
6 </Imageset>
7
```

Algoritmo 1: Definição de um elemento interativo tipo Check Box

Para facilitar o processo de mapeamento, pode ser utilizado o **CEImagesetEditor** [2006], como mostra a figura 3.

Figura 3: CEImagesetEditor – editor de mapeamento de widgets

4.3. Criando o arquivo *LooknFeel*

O arquivo *LooknFeel* possui a definição de como os *widgets* irão se comportar em resposta a interação do usuário, atribuindo ao *widget* a aparência apropriada, de acordo com seu estado. Por exemplo, para o *widget Check Box*, é definido qual aparência será usada quando o elemento se encontra desativado, selecionado ou ativado, através da interação feita pelo usuário através do mouse.

O arquivo *LooknFeel* é um arquivo XML que usa a notação denominada *Falagard Skinning System*. Esse sistema consiste em um conjunto de aprimoramentos para a biblioteca base da CEGUI. Sua função é permitir a criação de *widgets* definidos pelo usuário, utilizando as funcionalidades dos *widgets* padrão da CEGUI, mudando-se apenas a sua aparência.

4.4. Criando o arquivo *Font*

No arquivo *Font* é definida a fonte que será usada em alguns dos *widgets* criados, bem como as propriedades padrão para a mesma e sua localização. Podem ser definidas quantas fontes forem necessárias e cada fonte deverá estar definida em seu próprio arquivo *Font*. A CEGUI suporta dois tipos de fontes: True Type e Bitmap Fonts.

Os atributos principais contidos no arquivo de definição de fontes são o nome da fonte (que será referenciado pela aplicação), o arquivo da fonte propriamente dito (*Filename*), o seu tipo (*Dynamic – True Type ou Static – Bitmap Fonts*), o tamanho padrão da fonte (*Size*), a resolução padrão (*NativeHorzRes* e *NativeVertRes*), se a fonte aparecerá no mesmo tamanho em qualquer resolução e se ou não é necessário aplicar anti-aliasing (bordas arredondadas - *AntiAlias*).

O algoritmo 2, representa a definição de uma fonte do tipo True Type:

```
1 <?xml version="1.0" ?>
2 <Font Name="BlueHighway-8" Filename="bluehigh.ttf"
  Type="Dynamic" Size="8" NativeHorzRes="1024"
  NativeVertRes="768" AutoScaled="true" AntiAlias="true" />
```

Algoritmo 2: Definição de uma fonte

4.5. Criando o arquivo *Layout*

Os arquivos de *Layout* (também escritos em XML) descrevem o conjunto de *widgets* que serão criados, suas posições e atributos iniciais. O *Layout* define exatamente a aparência final que os *widgets* serão exibidos na tela para o usuário.

Para cada *widget* especificado como parte do layout, são definidos alguns atributos básicos como seu identificador usado para referenciá-lo (*ID*), o texto apresentado juntamente com o elemento (*Text*), qual a fonte utilizada no texto (*Font*) proveniente do arquivo de definição de fontes e sua posição na tela.

Para facilitar a criação do *Layout*, a CEGUI possui um editor próprio para essa finalidade: **CELayoutEditor** [2006], como mostra a figura 4. Esse

editor permite ao desenvolvedor da GUI desenhar a interface em um método similar ao usado em ambientes RAD (*Rapid Application Development*).

Figura 4: CELayoutEditor – editor de Layouts

O algoritmo 3, mostra o resultado final do arquivo de *Layout* criado com o **CELayoutEditor**, exibindo dois *widgets* tipo *Check Box*:

```
<?xml version="1.0" encoding="UTF-8"?>
<GUILayout>
<Window Type="RPGEDU/Checkbox" Name="CB_Rain" >
<Property Name="ID" Value="1" />
<Property Name="Text" Value="Chuva ON/OFF" />
<Property Name="UnifiedAreaRect"
Value="{{0.134099,0.000000},{0.416971,0.000000},{0.91
7069,0.000000},{0.497614,0.000000}}" />
<Property Name="UnifiedMaxSize"
Value="{{1.000000,0.000000},{1.000000,0.000000}}" />
</Window>
<Window Type="RPGEDU/Checkbox" Name="CB_Fog" >
<Property Name="Font" Value="Commonwealth-10" />
<Property Name="Text" Value="Neblina ON/OFF" />
<Property Name="UnifiedAreaRect"
Value="{{0.137413,0.000000},{0.488208,0.000000},{0.93
9613,0.000000},{0.574576,0.000000}}" />
<Property Name="UnifiedMaxSize"
Value="{{1.000000,0.000000},{1.000000,0.000000}}" />
</Window>
</GUILayout>
```

Algoritmo 3: Definição do arquivo de layout

4.6. Criando o arquivo *Scheme*

O arquivo *Scheme* é onde todos os arquivos de definição descritos anteriormente são unidos para formar o esquema completo da GUI, como mostra a figura 5, que será apresentado para o usuário final. Ele contém a definição de um conjunto de controles para a GUI. Esse arquivo é o mais utilizado na aplicação pois, ao carregado, exibe os *widgets* prontos para interação entre o usuário e a aplicação.

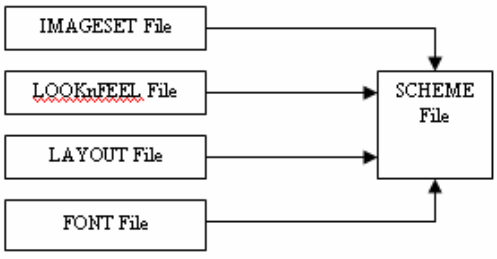

Figura 5: União dos arquivos de definição da GUI

O algoritmo 4, mostra o conteúdo do arquivo de esquema e como todos os outros arquivos de definição são unidos e carregados para formar o esquema completo da GUI:

```
<?xml version="1.0" ?>
<GUIScheme Name="RPGEDUscheme">
    <Imageset Name="RPGEDU"
    Filename="RPGEDU.imageset" />
    <Font Name="BlueHighway-8"
    Filename="bluehighway-8.font" />
    <LookNFeel Filename="RPGEDU.looknfeel" />
    <WindowSet Filename="CEGUIFalagardBase" />
    <FalagardMapping WindowType="RPGEDU/Button"
    TargetType="Falagard/Button"
    LookNFeel="RPGEDU/Button" />
    <FalagardMapping WindowType="RPGEDU/Checkbox"
    TargetType="Falagard/Checkbox"
    LookNFeel="RPGEDU/Checkbox" />
</GUIScheme>
```

Algoritmo 4: Definição do esquema

5. Conclusão

Apesar da complexidade em se trabalhar com diversos arquivos de definições distintos, a CEGUI atende a todos os requisitos para a criação de uma GUI eficiente e funcional, principalmente para a inserção de elementos interativos em jogos de computador.

A diversidade dos *widgets* disponíveis, a liberdade de criação da aparência dos mesmos e a flexibilidade de controle e manipulação desses elementos interativos são os pontos fortes dessa promissora biblioteca. Outro ponto importante é a flexibilidade na escolha do renderer e, como as definições estão em arquivos XML, não há necessidade de recompilação do projeto no caso de alterações no modelo da GUI.

Entretanto, como se encontra em fase de desenvolvimento, existem ainda diversos *bugs* que foram encontrados durante a sua utilização, principalmente nos editores de layout e de mapeamento de imagens. A documentação também é insuficiente e exemplos práticos são difíceis de se encontrar.

Referências

GAME PROGRAMMING WIKI. Disponível em: http://gpwiki.org/index.php/Libraries#GUI [Acessado em 02 agosto 2006]

CEGUI. Disponível em: http://www.cegui.org.uk [Acessado em 12 Agosto 2006]

CEIMAGESETEDITOR. Disponível em: http://www.cegui.org.uk/phpBB2/viewtopic.php?t=1346 [Acessado em 18 Junho 2006]

CELAYOUTEDITOR. Disponível em: http://www.cegui.org.uk/wiki/index.php/The_"official"_layout_editor [Acessado em 18 Junho 2006]

GTK+. Disponível em: http://gtk.org [Acessado em 25 setembro 2006]

GUINCHAN. Disponível em: http://guichan.sourceforge.net/index.shtml [Acessado em 25 setembro 2006]

GUIMP. Disponível em: http://www.gimp.org [Acessado em 25 setembro de 2006]

MW. Disponível em: http://themanaworld.org/ [Acessado em 26 setembro de 2006].

OGRE. Disponível em: http://www.ogre3d.org [Acessado em 06 outubro 2005]