SWING COMPONENTS

Even though the focus of this book is on Java 3D, it is not possible to ignore the use of Swing components completely. This is because, ultimately, a Java 3D virtual world has to be displayed onto the screen using these. However, through using the ViewManager, a custom utility class, this distraction, or burden can be significantly relieved, resulting in codes that are more easily understandably.

ViewManager places each of the Canvas3D inside a JPanel with the title of the Canvas3D put at the border of that panel. If a Graphical User Interface (GUI) is required, ViewManager can be invoked to add a sub-JPanel into the panel that holds the Canvas3D. The layout within each of the Canvas3D panels can be performed using BorderLayout, while the whole applet can be organized using GridBagLayout.

Figure 1 shows how ViewManager can be instantiated as well as an example.

If, in the example in Figure 1, the number of Canvas3D objects to be displayed is not equal to totalRows multiplied by totalColumns, ViewManager will use the value of totalColumns and wrap the Canvas3D objects accordingly. That is, if View Manager is requested to handle objects in a 1-row by 3-column fashion but there are four Canvas3D objects, the
Figure 1. ViewManager constructor and example

```java
ViewManager (JApplet jApplet, int totalRows, int totalColumns);
// jApplet - the JApplet that ViewManager should manage
// totalRows - number of rows of Canvas3D objects to be displayed in JApplet
// totalColumns - number of columns of Canvas3D objects to be displayed in JApplet

ViewManager viewManager = new ViewManager(this, 2, 2);
// Current applet to display 4 Canvas3D objects in a 2 by 2 configuration
```

The first three objects will be positioned in one row whereas the fourth one will be in the second row below the first Canvas3D object.

## ADDING Canvas3D

Figure 2 summarizes the usage and format of methods for adding Canvas3D objects.

Note that the methods shown can be broadly divided into three different groups. The first group adds Canvas3D objects without specifying locations, relying instead on ViewManager

Figure 2. Methods for adding Canvas3D objects

```java
viewManager.add (Canvas3D canvas3D, String title);
viewManager.add (Canvas3D canvas3D, String title, JPanel JPanel);
viewManager.add (Canvas3D canvas3D, String title, JPanel JPanel, String location);
viewManager.add (Canvas3D canvas3D, String title, int index);
viewManager.add (Canvas3D canvas3D, String title, JPanel JPanel, int index);
viewManager.add (Canvas3D canvas3D, String title, JPanel JPanel, String location, int index);
viewManager.add (Canvas3D canvas3D, String title, int row, int column);
viewManager.add (Canvas3D canvas3D, String title, JPanel JPanel, int row, int column);
viewManager.add (Canvas3D canvas3D, String title, JPanel JPanel, String location, int row, int column);
// canvas3D - Canvas3D to be added into the applet
// title - title to identify the Canvas3D
// JPanel - additional panel for holding, say, the Panel1 controls, to be added into the Panel1
// location – may be ViewManager.TOP_CONTROL, ViewManager.RIGHT_CONTROL,
// ViewManager.BOTTOM_CONTROL or ViewManager.LEFT_CONTROL for specifying
// the position for adding JPanel with respect to the Canvas3D
// row - specify the row for adding the Canvas3D
// column - specify the column for adding the Canvas3D
// index - specify the index of the Canvas3D to be added
```
to place the objects automatically. The second group is based on the index of the Canvas3D object to being added. Using this, we can control the sequence of the Canvas3D objects to be added into the applet. The last group allows complete control on where Canvas3D should be added, through the row and column of the position of the Canvas3D in the applet.

In the methods for each group, the first one allows one to add just the Canvas3D and its title. The second one has the additional ability to add a JPanel into the panel that holds the Canvas3D, with a default position that is below the Canvas3D. With the third method in each group, we can even specify the position of the JPanel with respect to the Canvas3D.

Most applications will however use just the first method, as ViewManager can also be used to create the control panel. Nevertheless, the other two methods can also be invoked to construct a JPanel manually if the needs arise. Note that all of the methods will return the JPanel that holds the Canvas3D. This gives the handle for any further modifications to be made.

As a usage example, if viewManager is an instantiated object of ViewManager, and there is a need to add a Canvas3D canvas3D with the title “test,” we can use the method

```java
viewManager.add(canvas3D, "test");
```

This can be repeated to add other Canvas3D objects into the applet. ViewManager will automatically position the Canvas3D, starting from the top left corner of the applet.

## ADDING CONTROL PANEL

As previously mentioned, ViewManager allows for the addition of a control panel for each Canvas3D. However, instead of performing this through the constructor, the SetControlPanel method illustrated in Figure 3 can also be used for this purpose.

Note that since ViewManager will only help in positioning, the control panel must be fully programmed before it is added to the Canvas3D.

![Figure 3. setControlPanel method of ViewManager](image)

```java
void viewManager.setControlPanel(JFrame jPanel);
void viewManager.setControlPanel(JFrame jPanel, String location);
// jPanel - panel to be added to the most recently added Canvas3D
// location – may be ViewManager.TOP_CONTROL, ViewManager.RIGHT_CONTROL,
// ViewManager.BOTTOM_CONTROL or ViewManager.LEFT_CONTROL for
// specifying the position for adding JPanel with respect to the Canvas3D
```
Yet another way is add a control panel is to use ViewManager to create an empty panel first and then add the controls later. The method that can be used for this purpose is illustrated in Figure 4.

Invoking createControlPanel() without any parameter will create a control panel at defaults given by location = ViewManager.BOTTOM_CONTROL, row = 1, and column = 3. Although this default is only for 3 controls in a single row, more controls can in fact be added through the usual wrapping process.

Note however that the setControlPanel and createControlPanel methods are applicable only to the last Canvas3D added. Also, each Canvas3D can only have one control panel. Thus, invoking one of these methods will overwrite any control panel that has been set or created earlier.

**Figure 4. createControlPanel method of ViewManager**

```java
JPanel viewManager.createControlPanel();
JPanel viewManager.createControlPanel(int row, int column);
JPanel viewManager.createControlPanel(String location, int row, int column);
// location – May be ViewManager.TOP_CONTROL, ViewManager.RIGHT_CONTROL, ViewManager.BOTTOM_CONTROL or ViewManager.LEFT_CONTROL
// specifying the position for the JPanel to be added with respect to the Canvas3D
// row - total number of rows of controls to be added
// column - total number of columns of the controls to be added
```

**Figure 5. Using ViewManager correctly**

1. viewManager.add(canvas3D1, “canvas3D1”);
2. viewManager.createControlPanel();
3. viewManager.add(canvas3D2, “canvas3D2”);
4. viewManager.createControlPanel();

**Figure 6. Using ViewManager incorrectly**

1. viewManager.add(canvas3D1, “canvas3D1”);
2. viewManager.add(canvas3D2, “canvas3D2”);
3. viewManager.createControlPanel();
4. viewManager.createControlPanel();

Copyright © 2009, IGI Global, distributing in print or electronic forms without written permission of IGI Global is prohibited.
As an example, Figure 5 shows how two Canvas3D, canvas3D1 and canvas3D2 with a control panel each, can be added to viewManager, an instantiated object of ViewManager.

If the code sequence is in the wrong order as illustrated in Figure 6, an error will result. Specifically, this code segment will add canvas3D1 without any control panel, while canvas3D2 will have only one control panel. This is despite the two attempts to create two control panels, with the second one overwriting the first one.

**ADDING CONTROLS**

Since ViewManager only helps us to position the controls for a Canvas3D object, the creation of these controls and their ActionListener have to be carried out manually. Note that, in this process, we need to invoke the createControlPanel method first. Specifically, the addControl methods illustrated in Figure 7 can only be used in conjunction with the appropriate createControlPanel method. The setControlPanel method cannot be used for the addition of controls.

*Figure 7. addControl method summary*

```java
JComponent viewManager.addControl(JComponent |Component);  
JComponent viewManager.addControl(JComponent |Component, int height, int width);  
JComponent viewManager.addControl(JComponent |Component, int index);  
JComponent viewManager.addControl(JComponent |Component, int index, int height, int width);  
JComponent viewManager.addControl(JComponent |Component, int row, int column, int height,  
  // int width);  
// |Component – an object from JComponent or any of its subclasses to be added  
// index - index of the |Component with respect to other |Component objects  
// row - row for adding |Component, with respect to other |Component objects  
// column - column for adding |Component, with respect to other |Component objects  
// height - number of rows |Component would stretch across  
// width - number of columns |Component would stretch across
```

*Figure 8. Using addControl*

1. viewManager.addControl (title, 1, 3);  
2. viewManager.addControl (minus);  
3. viewManager.addControl (value);  
4. viewManager.addControl (plus);
JComponent has many subclasses and two or more commonly used ones are JLabel and JButtons, the usage of which are given in the Java API.

As an example, suppose viewManager is an instantiated object of ViewManager, a Canvas3D object has been added, and an createControlPanel method has been invoked. Then, the code segment in Figure 8 will add a JLabel title that stretches over three columns, a JButton minus, a JLabel value, and a JButton plus. Specifically, ViewManager will position the JLabel title in the first row and have it stretch over three columns, and place the other controls in the second row, in the order of minus, value, and plus.

GETTING OPTIMUM WIDTH AND HEIGHT

The optimum width and height of the applet, or the width and height without stretching, can be obtained by using the methods in Figure 9.

These methods are particularly useful when it is used together with the MainFrame utility class that allows applets to be run as applications. An example code segment is provided in Figure 10. The lines are from the main method of ViewModelApp, which uses the MainFrame utility class to enable it to be run as an application. Here, ViewManager is called using viewModel.viewManager because only the ViewManager object that has been used to manage the application layout is able to return the correct width and height. A new instance of ViewManager created in the main method will not be able to return the correct width and height.

Figure 9. Methods for getting applet size

```java
int viewManager.getOptimumFrameWidth();
int viewManager.getOptimumFrameHeight();
```

Figure 10. ViewManager and Mainframe

```
1. ViewModelApp viewModel = new ViewModelApp();
2. Frame frame = new MainFrame( viewModel,
3.     viewModel.viewManager.getOptimumFrameWidth(),
4.     viewModel.viewManager.getOptimumFrameHeight() );
```
FRAME CENTERING

This method shown in Figure 11 places the frame in the centre of the screen, and can be used in conjunction with the MainFrame utility class. The line, which uses ViewManager to place the frame in the centre of the screen, is taken from the main method in View-ModelApp. The object frame is an instance of MainFrame that holds the viewModel, an instance of ViewModel.

GETTING CONFIGURATION FOR Canvas3D

Figure 12 illustrates the method that can be used to return the GraphicsConfiguration that can be used for Canvas3D. Note that a warning will be given if Canvas3D is instantiated using null for its GraphicsConfiguration.

Figure 11. placeInScreenCenter method

```java
viewModel.viewManager.placeInScreenCenter(frame);
```

Figure 12. GraphicsConfiguration

```java
1. GraphicsConfiguration config = viewManager.getPreferredGraphicsConfiguration();
2. Canvas3D canvas3D = new Canvas3D(config);
```