An object model for object-oriented environments. In one embodiment, a system including a plurality of container objects, each container object having a data object, a controller object, and a displayer object. The data object stores data. The controller object provides at least one interaction with the data of the data object. The displayer object provides at least one response for the data of the data object, via the controller object.
FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>AL</td>
<td>Albania</td>
<td>ES</td>
<td>Spain</td>
<td>LS</td>
<td>Lesotho</td>
<td>SI</td>
<td>Slovenia</td>
</tr>
<tr>
<td>AM</td>
<td>Armenia</td>
<td>FI</td>
<td>Finland</td>
<td>LT</td>
<td>Lithuania</td>
<td>SK</td>
<td>Slovakia</td>
</tr>
<tr>
<td>AT</td>
<td>Austria</td>
<td>FR</td>
<td>France</td>
<td>LU</td>
<td>Luxembourg</td>
<td>SN</td>
<td>Senegal</td>
</tr>
<tr>
<td>AU</td>
<td>Australia</td>
<td>GB</td>
<td>United Kingdom</td>
<td>LV</td>
<td>Latvia</td>
<td>SZ</td>
<td>Swaziland</td>
</tr>
<tr>
<td>AZ</td>
<td>Azerbaijan</td>
<td>GE</td>
<td>Georgia</td>
<td>MC</td>
<td>Monaco</td>
<td>TD</td>
<td>Chad</td>
</tr>
<tr>
<td>BA</td>
<td>Bosnia and Herzegovina</td>
<td>GH</td>
<td>Ghana</td>
<td>MD</td>
<td>Republic of Moldova</td>
<td>TG</td>
<td>Togo</td>
</tr>
<tr>
<td>BB</td>
<td>Barbados</td>
<td>GN</td>
<td>Guinea</td>
<td>MG</td>
<td>Madagascar</td>
<td>TJ</td>
<td>Tajikistan</td>
</tr>
<tr>
<td>BE</td>
<td>Belgium</td>
<td>GR</td>
<td>Greece</td>
<td>MK</td>
<td>The former Yugoslav</td>
<td>TM</td>
<td>Turkmenistan</td>
</tr>
<tr>
<td>BF</td>
<td>Burkina Faso</td>
<td>GE</td>
<td>Georgia</td>
<td>ML</td>
<td>Mali</td>
<td>TR</td>
<td>Turkey</td>
</tr>
<tr>
<td>BG</td>
<td>Bulgaria</td>
<td>HU</td>
<td>Hungary</td>
<td>MN</td>
<td>Mongolia</td>
<td>TT</td>
<td>Trinidad and Tobago</td>
</tr>
<tr>
<td>RJ</td>
<td>Benin</td>
<td>IE</td>
<td>Ireland</td>
<td>MR</td>
<td>Mauritania</td>
<td>UA</td>
<td>Ukraine</td>
</tr>
<tr>
<td>BR</td>
<td>Brazil</td>
<td>IL</td>
<td>Israel</td>
<td>MW</td>
<td>Malawi</td>
<td>UG</td>
<td>Uganda</td>
</tr>
<tr>
<td>BY</td>
<td>Belarus</td>
<td>IS</td>
<td>Iceland</td>
<td>MX</td>
<td>Mexico</td>
<td>US</td>
<td>United States of America</td>
</tr>
<tr>
<td>CA</td>
<td>Canada</td>
<td>IT</td>
<td>Italy</td>
<td>NE</td>
<td>Niger</td>
<td>UZ</td>
<td>Uzbekistan</td>
</tr>
<tr>
<td>CF</td>
<td>Central African Republic</td>
<td>JP</td>
<td>Japan</td>
<td>NL</td>
<td>Netherlands</td>
<td>VN</td>
<td>Viet Nam</td>
</tr>
<tr>
<td>CG</td>
<td>Congo</td>
<td>KE</td>
<td>Kenya</td>
<td>NO</td>
<td>Norway</td>
<td>YU</td>
<td>Yugoslavia</td>
</tr>
<tr>
<td>CH</td>
<td>Switzerland</td>
<td>KG</td>
<td>Kyrgyzstan</td>
<td>NZ</td>
<td>New Zealand</td>
<td>ZW</td>
<td>Zimbabwe</td>
</tr>
<tr>
<td>CI</td>
<td>Côte d'Ivoire</td>
<td>KP</td>
<td>Democratic People's Republic of Korea</td>
<td>PL</td>
<td>Poland</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CM</td>
<td>Cameroon</td>
<td>KR</td>
<td>Republic of Korea</td>
<td>PT</td>
<td>Portugal</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CN</td>
<td>China</td>
<td>KZ</td>
<td>Kazakhstan</td>
<td>RO</td>
<td>Romania</td>
<td>RU</td>
<td>Russian Federation</td>
</tr>
<tr>
<td>CU</td>
<td>Cuba</td>
<td>LC</td>
<td>Saint Lucia</td>
<td>SD</td>
<td>Sudan</td>
<td>SE</td>
<td>Sweden</td>
</tr>
<tr>
<td>CZ</td>
<td>Czech Republic</td>
<td>LI</td>
<td>Liechtenstein</td>
<td>SG</td>
<td>Singapore</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DE</td>
<td>Germany</td>
<td>LK</td>
<td>Sri Lanka</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DK</td>
<td>Denmark</td>
<td>LR</td>
<td>Liberia</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EE</td>
<td>Estonia</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Notes:**
- Codes are used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.
- The table lists countries and their codes, with the codes representing the country's ISO 3166-1 alpha-3 code, except for a few countries where a different code is used.
OBJECT MODEL FOR OBJECT-ORIENTED COMPUTING ENVIRONMENTS

RELATED APPLICATIONS

This application is related to the following applications, which are hereby incorporated by reference: Dynamic Data Cache for Object-Oriented Computing Environments, [SLWK docket 777.171US1] and Dynamic Object Behavior for Object-Oriented Computing Environments [SLWK docket 777.172US1].

FIELD OF THE INVENTION

This invention relates generally to object-oriented computing environments, and more particularly to an object model for such environments.

BACKGROUND OF THE INVENTION

Object-oriented programming environments are currently the standard environment in which computer programs are developed. For example, within the Microsoft Windows operating system, programs may be developed using the Component Object Model (COM) architecture. Object-oriented programming environments provide a modular manner by which developers can develop complex and sophisticated computer programs.

Generally, an object may include data and methods by which that data is accessed and changed. Thus, new methods may be added to the object for accessing and changing the data in other ways not previously possible, without affecting existing methods, and the external objects that rely on these existing methods. Upon receiving a message, or in response to an event, an object typically executes a particular method, resulting in the data within that object being retrieved or changed.
SUMMARY OF THE INVENTION

The invention provides for a novel object model for object-oriented environments. In one embodiment, a system includes a plurality of container objects. Each container object includes a data object, a controller object, and a displayer object. The data object stores data; in one specific embodiment, via a plurality of static properties. The controller object provides at least one interaction with the data of the data object; in one specific embodiment, each interaction is a message or an event. The displayer object provides at least one response for the data of the data object, via the controller object; in one specific embodiment, each response is a visual representation of the data, or a non-visual representation of the data.

The object model of the invention provides for advantages not found in the prior art. The object model enables software developers to more easily create, maintain, and enhance applications. For example, in the prior art, if a new visual look for a software application (i.e., have the buttons, check boxes, etc., appear visually different on the screen) is desired, generally one or more central dynamically linked libraries (DLL) may be required to be changed, such that each software application must be extensively tested for compatibility therewith. However, under an embodiment of the invention, only a new displayer object would have to be added to components of the software application, such that incompatibility with other software applications would not result.

The invention includes systems, methods, computers, and computer-readable media of varying scope. Besides the embodiments, advantages and aspects of the invention described here, the invention also includes other embodiments, advantages and aspects, as will become apparent by reading and studying the drawings and the following description.
BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 shows a diagram of the hardware and operating environment in conjunction with which embodiments of the invention may be practiced;

FIG. 2 shows a block diagram of a system according to one embodiment of the invention;

FIG. 3 shows a diagram of a button control according to an embodiment of the invention; and,

FIG. 4 shows a diagram of a check box control according to an embodiment of the invention.

DETAILED DESCRIPTION

In the following detailed description of exemplary embodiments of the invention, reference is made to the accompanying drawings which form a part hereof, and in which is shown by way of illustration specific exemplary embodiments in which the invention may be practiced. These embodiments are described in sufficient detail to enable those skilled in the art to practice the invention, and it is to be understood that other embodiments may be utilized and that logical, mechanical, electrical and other changes may be made without departing from the spirit or scope of the present invention. The following detailed description is, therefore, not to be taken in a limiting sense, and the scope of the present invention is defined only by the appended claims.

The detailed description is divided into seven sections. In the first section, the hardware and the operating environment in conjunction with which embodiments of the invention may be practiced are described. In the second section, a system of one embodiment of the invention is presented, with specific recitation of a container object of the system. In the third section, a data object of a container object in accordance with an embodiment of the invention is described in more detail. In the fourth section, a data cache object of a container object in accordance with an embodiment of the invention is described in more detail. In the fifth section, a controller object in accordance with an embodiment of the invention is described in more detail. In the sixth section, a displayer
object in accordance with an embodiment of the invention is described in more
detail. Finally, in the seventh section, a conclusion of the detailed description is
provided.

Hardware and Operating Environment

Referring to FIG. 1, a diagram of the hardware and operating
environment in conjunction with which embodiments of the invention may be
practiced is shown. The description of FIG. 1 is intended to provide a brief,
general description of suitable computer hardware and a suitable computing
environment in conjunction with which the invention may be implemented.
Although not required, the invention is described in the general context of
computer-executable instructions, such as program modules, being executed by a
computer, such as a personal computer. Generally, program modules include
routines, programs, objects, components, data structures, etc., that perform
particular tasks or implement particular abstract data types.

Moreover, those skilled in the art will appreciate that the invention may
be practiced with other computer system configurations, including hand-held
devices, multiprocessor systems, microprocessor-based or programmable
consumer electronics, network PCS, minicomputers, mainframe computers, and
the like. The invention may also be practiced in distributed computing
environments where tasks are performed by remote processing devices that are
linked through a communications network. In a distributed computing
environment, program modules may be located in both local and remote memory
storage devices.

The exemplary hardware and operating environment of FIG. 1 for
implementing the invention includes a general purpose computing device in the
form of a computer 20, including a processing unit 21, a system memory 22, and
a system bus 23 that operatively couples various system components include the
system memory to the processing unit 21. There may be only one or there may
be more than one processing unit 21, such that the processor of computer 20
comprises a single central-processing unit (CPU), or a plurality of processing
units, commonly referred to as a parallel processing environment. The computer 20 may be a conventional computer, a distributed computer, or any other type of computer; the invention is not so limited.

The system bus 23 may be any of several types of bus structures including a memory bus or memory controller, a peripheral bus, and a local bus using any of a variety of bus architectures. The system memory may also be referred to as simply the memory, and includes read only memory (ROM) 24 and random access memory (RAM) 25. A basic input/output system (BIOS) 26, containing the basic routines that help to transfer information between elements within the computer 20, such as during start-up, is stored in ROM 24. The computer 20 further includes a hard disk drive 27 for reading from and writing to a hard disk, not shown, a magnetic disk drive 28 for reading from or writing to a removable magnetic disk 29, and an optical disk drive 30 for reading from or writing to a removable optical disk 31 such as a CD ROM or other optical media.

The hard disk drive 27, magnetic disk drive 28, and optical disk drive 30 are connected to the system bus 23 by a hard disk drive interface 32, a magnetic disk drive interface 33, and an optical disk drive interface 34, respectively. The drives and their associated computer-readable media provide nonvolatile storage of computer-readable instructions, data structures, program modules and other data for the computer 20. It should be appreciated by those skilled in the art that any type of computer-readable media which can store data that is accessible by a computer, such as magnetic cassettes, flash memory cards, digital video disks, Bernoulli cartridges, random access memories (RAMs), read only memories (ROMs), and the like, may be used in the exemplary operating environment.

A number of program modules may be stored on the hard disk, magnetic disk 29, optical disk 31, ROM 24, or RAM 25, including an operating system 35, one or more application programs 36, other program modules 37, and program data 38. A user may enter commands and information into the personal computer 20 through input devices such as a keyboard 40 and pointing device 42. Other input devices (not shown) may include a microphone, joystick, game pad, satellite dish, scanner, or the like. These and other input devices are often
connected to the processing unit 21 through a serial port interface 46 that is
coupled to the system bus, but may be connected by other interfaces, such as a
parallel port, game port, or a universal serial bus (USB). A monitor 47 or other
type of display device is also connected to the system bus 23 via an interface,
such as a video adapter 48. In addition to the monitor, computers typically
include other peripheral output devices (not shown), such as speakers and
printers.

The computer 20 may operate in a networked environment using logical
connections to one or more remote computers, such as remote computer 49.
These logical connections are achieved by a communication device coupled to or
a part of the computer 20; the invention is not limited to a particular type of
communications device. The remote computer 49 may be another computer, a
server, a router, a network PC, a client, a peer device or other common network
node, and typically includes many or all of the elements described above relative
to the computer 20, although only a memory storage device 50 has been
illustrated in FIG. 1. The logical connections depicted in FIG. 1 include a local-
area network (LAN) 51 and a wide-area network (WAN) 52. Such networking
environments are commonplace in office networks, enterprise-wide computer
networks, intranets and the Internet, which are all types of networks.

When used in a LAN-networking environment, the computer 20 is
connected to the local network 51 through a network interface or adapter 53,
which is one type of communications device. When used in a WAN-networking
environment, the computer 20 typically includes a modem 54, a type of
communications device, or any other type of communications device for
establishing communications over the wide area network 52, such as the Internet.
The modem 54, which may be internal or external, is connected to the system
bus 23 via the serial port interface 46. In a networked environment, program
modules depicted relative to the personal computer 20, or portions thereof, may
be stored in the remote memory storage device. It is appreciated that the
network connections shown are exemplary and other means of and
communications devices for establishing a communications link between the computers may be used.

The hardware and operating environment in conjunction with which embodiments of the invention may be practiced has been described. The computer in conjunction with which embodiments of the invention may be practiced may be a conventional computer, a distributed computer, or any other type of computer; the invention is not so limited. Such a computer typically includes one or more processing units as its processor, and a computer-readable medium such as a memory. The computer may also include a communications device such as a network adapter or a modem, so that it is able to communicatively couple other computers.

System and Container Object

In this section of the detailed description, a description of a system, and specifically a container object of the system, according to an embodiment of the invention is provided. The description is provided by reference to FIG. 2. Referring now to FIG. 2, a system according to an embodiment of the invention is shown. The system includes a container object 200 (also referred to as a control, as opposed to a controller), and one or more external objects 210. The container object 202 includes a data object 202, a data cache object 204, one or more controller objects 206, and one or more displayer objects 208. Each of these objects can in one embodiment be a Component Object Model (COM) object, for use in programming in conjunction with the Microsoft Windows operating system, although the invention is not so limited.

The container object 200 is a container in that it encompasses, or contains, all the data object 202, data cache object 204, the controller object 206, and the displayer objects 208. The container object 200 may be viewed in one embodiment as an actual control or component; however, the invention is not so limited. The object 200 is a manager of the objects which it contains. The object 200, or control, is responsible for piecing the data object 202, the controller objects 206, and the displayer objects 208, to assemble a coherent,
usable object. The object 200 is also responsible for the data cache object 204. Thus, the object 200 maintains a list of the data object 202, the controller objects 206, and the displayers objects 208; provides for the data cache object 204; and provides the ability to add and remove controller objects 206 and displayers objects 208.

In one embodiment of the invention, as can be appreciated by those of ordinary skill within the art, the control -- that is, the container object -- has an interface specified in C++ as follows:

```cpp
public interface Icontrol
{
    public Idata getData();

    public I��sDisplayCollection getDisplayers();
    public I��sControllerCollection getControllers();
}
```

The external objects 210 are objects that communicate with the container object 200. The external objects 210 are defined primarily as being external to the container object 200. These objects 210 may, for example, be other container objects, although the invention is not so limited. As will be described, external objects 210 communicate in one embodiment with a container object 200 via the controller object 206 thereof. That is, the objects 210 are communicatively coupled to the container object 200 via the controller object 206.

It is noted that as to the embodiments of the invention described in conjunction with FIG. 2, the objects can in one embodiment be stored on a computer-readable storage medium, such as a floppy disk or a compact-disc read-only-memory (CD-ROM). In another embodiment, the objects are represented as data residing in a memory of a computer, executed by the processor thereof. The invention is not particularly limited, however.
Data Object

In this section of the detailed description, a data object of a container object according to an embodiment of the invention is described. The data object may be a data object such as the data object 202 of the container object 200 of FIG. 2. The invention is not so limited.

Referring to FIG. 2, data object 202 is the object that stores the data of the container object 200. In one embodiment, such data is stored as properties. The object 204 stores static properties, which are data that is always associated with the object 202 (as opposed to dynamic properties, as will be described). In one embodiment of the invention, there can be more than one data object 202.

The data object 202 is thus an object that represents the data to be displayed by a displayer object 208, as will be described, although the word displayed as used herein does not necessarily connotate a visual display; it can also be a non-visual display. The data in one embodiment can be an instance of a class that represents the data most efficiently.

In one embodiment, a feature of a data object is that it notifies other, dependent data objects whenever one or more of its variables are changed. Thus, sub-classes of the data object can inherit a changed method which, when invoked, causes the change notification to be sent to each associated controller object. Furthermore, in one embodiment, the only mechanism by which the values of the private variables of a data object can be changed is through the properties of the data object, that is, methods. These methods are desirably written such that they send a change notification when called.

As shown in FIG. 2, the data object 202 has no direct reference to the controller object 206, or the displayer object 208, but rather communicates with the controller object 206 via the data cache object 204, as will be described, and communicates with the displayer object 208 via the controller object 206 (and thus itself via the data cache object 204). It is noted, however, that the invention does not require a data cache object 204; in such an embodiment, the controller object 206 directly communicates with the data object 202. That is, the
controller object talks directly to the data object if the data cache object is not present.

In one embodiment, the data object 202 contains a series of setter and getter methods that allow another object to set the value of a piece of information contained by the data object 202. When the set method is called, it fires a notification to all registered objects. If the return value from the notification is false, then the data does not change. Otherwise, the data object sets the value and sends another notification to all the registered objects.

In one embodiment of the invention, as can be appreciated by those of ordinary skill within the art, the data object -- has an interface specified in C++ as follows:

```cpp
public interface IData
{
    public void addDataNotification(IDataNotification notify);
    public void removeDataNotification(IDataNotification notify);
    public boolean notifyBeforeChange(ChangeNotification n);
    public void notifyAfterChange(ChangeNotification n);
}
```

As can also be understood by those of ordinary skill in the art, an example in C++ pseudo code to implement a data object that fires data change notifications is as follows:

```cpp
public class ExampleData extends windows.ui.base.Data
{
    private String name;

    public void setName(String n)
    {
        ChangeNotification cn = new ChangeNotification(this.name, n);
        if (notifyBeforeChange(cn))
        {
            this.name = n;
            notifyAfterChange(cn);
        }
    }
```
Furthermore, when data notifications are created and sent to objects, the object passed around is a change-notification object, which contains the old value and the new value. By passing both values, the controller object is able to determine if any action is made.

In one embodiment of the invention, as can be appreciated by those of ordinary skill within the art, the data notification and the change notification have interfaces specified in C++ as follows:

```java
public interface IDataNotification extends windows.system.Event
{
    public boolean onBeforeChange(ChangeNotification n);
    public void onAfterChange(ChangeNotification n);
}

public interface IChangeNotification
{
    public void setOldValue(Object oldvalue);
    public Object getOldValue();

    public void setNewValue(Object newvalue);
    public Object getNewValue();

    public void setDataName();
    public String getDataName();
}
```

Thus, there are in one embodiment four different methods that are operable as to a data object. First, an add data notification method, which registers the object that wants to receive data change notifications. Second, a remove data notification method, which removes the notification sink object from the list of items that receives notification when a data member changes. Third, a notify before change method that is called before the actual change to the data is made; if a value of false is returned by the sink object, then the
changes do not occur. Finally, fourth, a notify after change method which is called after the actual data has been changed.

To receive notifications when the data changes, an object that is interested in receiving notifications registers itself with the data object via the add data notification method. This sets up communication between the object and the data. Thus, to receive notification, the object desirably should have implemented a data notification interface, having two methods. The first method is a on before change method, called before the actual change occurs; if the data is valid, a true value is returned, otherwise false is returned. The second method is an on after change method, which is called when the change to the data has been effectuated.

As can be understood by those of ordinary skill in the art, an example in C++ pseudo code to implement an object that sinks data change notifications is as follows:

```java
public class ReceiveDataNotification implements IDataNotification {
    private Data data = new Data();

    public ReceiveDataNotification() {
        data.addDataNotification(this);
    }

    public void finalize() {
        data.removeDataNotification(this);
    }

    public boolean OnBeforeChange(ChangeNotification cn) {
        String thevalue = (String)cn.getValue();
        if (thevalue.equals(FooBar))
            return false;

        return true;
    }
}
```
public void onAfterChange(ChangeNotification cn)
{
    windows.ui.MessageBox mb = new windows.ui.MessageBox();
    mb.show(cn.getNewValue());
}

It is noted that the invention is not limited to the terminology set forth in
this application, and that alternative terminology may also be used, but still
encompass the invention. For example, in one embodiment, an alternative
terminology of C-MIR is used: control, model, interaction and response.
Whereas elsewhere in this application the term container is used, in this
alternative embodiment the term control is used. Furthermore, where in this
application the term data is used, in this alternative embodiment the term model
is used. Also, where in this application the term controller is used, in this
alternative embodiment the term interaction is used. Finally, where in this
application the term display is used, in this alternative embodiment the term
response is used. Thus, the invention as claimed covers these alternative
embodiments; that is, the term container encompasses the term control, the term
data encompasses the term model, the term controller encompasses the term
interaction, and the term display encompasses the term response.

Data Cache Object

In this section of the detailed description, a data cache object of a
container object according to an embodiment of the invention is described. The
data cache object may be a data cache object such as the data cache object 204 of
the container object 200 of FIG. 2. The invention is not so limited.

Referring to FIG. 2, the data cache object 204 is communicatively
coupled to the data object 202 and the controller object 206. The data cache
object 204 thus in one embodiment is the only manner through which access is
able to be made to the data stored in the data object 202. That is, the data cache
object 204 is such that all access to the static properties of the object 202 is made
through object 204. The data cache object directly governs all access to the data object.

In addition, the data cache object 204 temporarily stores any dynamic properties for the data object 202 that may have been requested to be added to the data object 202 by other objects. Dynamic properties are data that is not permanently associated with the definition of data object 202, as opposed to static properties that always are. Thus, the data cache object 204 is a mechanism that allows another object to store state and other information, in the form of dynamic properties, that may be of interested to itself and other objects. For example, an object may add a text color property to an object that does not currently have this data. When another objects asks for the properties of this object, it then is able to see that the text color property exists, and to use this information.

The data cache object may be in one embodiment of the invention a dynamic data cache object, as described in the application previously incorporated by reference, Dynamic Data Cache for Object-Oriented Computing Environments [SLWK docket 777.171US1]. The invention is not so limited, however.

In one embodiment of the invention, as can be appreciated by those of ordinary skill within the art, the data cache object has an interface specified in C++ as follows:

```cpp
public interface IDataCache
{
    public void setValue(String datamember, Object value);
    public Object getValue(String datamember);
    public boolean isRead(String datamember);
    public boolean isWrite(String datamember);
    public boolean isReadWrite(String datamember);
    public boolean isHidden(String datamember);
    public void setDefeferCommit(boolean defer);
    public boolean getDefeferCommit();
}
```
public void addItem(String datamember);
public void removeItem(String datamember);

public void clearCache();
public void commitChanges();
}

Controller Object

In this section of the detailed description, a controller object of a container object according to an embodiment of the invention is described. The controller object may be a controller object such as the controller object 206 of the container object 200 of FIG. 2. The invention is not so limited.

Referring to FIG. 2, controller object 206 is communicatively coupled to the data cache object 204, the external objects 210, and the display objects 208. The controller object 206 is the object through which all accesses to the data cache object 204 are made, in one embodiment. In one embodiment, such accesses can be by messages sent by other objects, or events that have been preset by objects and that have been triggered. It is noted that the controller object 206 communicates indirectly to the data object 202 through the data cache object 204. Thus, the controller object at least indirectly governs all access to the data object. Furthermore, other objects, such as external objects 210 and display objects 208, communicate indirectly to the data cache object 204 through the controller object 206. However, in an embodiment where there is no data cache object, then such access to the data object is directly governed by the controller object. The controller object sets data in the data object, based on raw inputs and events. The controller object communicates with the data object directly (if there is no data cache object), or with the data cache object (if present).

When an event or message occurs, such as a raw input event, a controller is called on a handle-event interface of the object. This gives the controller
object the opportunity to process the event and act accordingly. Besides input notification, the controller is notified when data is changed. When this occurs, the controller decides whether a relevant displayer object should be notified, by an update method. In one embodiment, the displayer object is optimized such that it calls back a get-changed-data method of the controller object to determine what specific data member has actually been changed.

In one embodiment of the invention, as can be appreciated by those of ordinary skill within the art, the controller object has an interface specified in C++ as follows:

```java
public interface Icontroller extends Icontrol, IDataNotification
{
    public void handleEvent(Event event);
    public changeNotification getChangedData();
}
```

It is noted that in at least one embodiment of the invention, the controller object (or, interactions) are not the same interface to the data object. The controller object is able to set data in this embodiment, but other objects (such as displayer objects, or external objects) are also able to set the data. That is, the displayer objects are able to observe data changes without interaction with the controller objects themselves. The controller objects in this embodiment do not cause the displayer objects to change. Thus, the interaction and response are decoupled.

**Displayer Object**

In this section of the detailed description, a displayer object of a container object according to an embodiment of the invention is described. The displayer object may be a displayer object such as the displayer object 208 of the container object 200 of FIG. 2. The invention is not so limited.

Referring to FIG. 2, the displayer objects 208 are part of the container object 200. They are communicatively coupled to the controller object 206. The
displayer objects 208 can add new (dynamic) properties to the data object 202 by making a request of the controller object 206, which relays the message to the data cache object 204, as well as access dynamic or static properties, again by making an appropriate request to the controller object 206. In another embodiment, the displayer objects 208 are able to communicate with the data object 202 and the data cache object 204 directly, as is shown in FIG. 2, without going through the controller object 206.

The primary purpose of the displayer object is to display, either visually or non-visually, the data that is represented by the data object. A displayer responsibility is thus to display a particular piece of information. A displayer object does not have to display all of the information contained by the data. A particular visual or non-visual display of the data by a displayer object is called a response. A given displayer object may have multiple responses, both or either visual or non-visual responses. That is, the term “displayer” does not necessarily imply a visual response, but rather a response that may be visual or non-visual.

An example of a displayer object is shown in conjunction with FIG. 3. Referring to FIG. 3, a diagram of a button control is shown. The button control 300 has five different displayer objects: a border displayer 302 to display the border of the button control; a focus displayer 304 to display a dotted-line border when the control has focus (i.e., is the currently active control); a background displayer 306 to display the background of the control; a text displayer 308 to display the text of the control; and, a default action displayer 310 to indicate the default action of the control. Thus, the example of FIG. 3 illustrates that it takes many displayer objects to implement a simple button control. However, by having displayer objects that are focused on different tasks, it is thus easy to replace the displayer as the system (or the control) matures over time.

When data of the data object changes, a controller object is notified. The controller then determines whether it should alert the displayer objects associated with the control (i.e., the displayer objects of the container object). When a change is needed, the controller alerts a displayer object by a call to a draw method of the object. If a given displayer object is interested in which particular
property or data member that has specifically changed, it calls the controller
objects get changed data method, which returns a change notification. Thus, the
display object determines whether it should update its display based on the
data member that has actually changed.

Furthermore, in another embodiment of the invention, the display
object, once notified by a controller object, is able to directly interface with the
data object (if no data cache object is present), or the data cache object (if
present). That is, the display object and the controller object are decoupled. In
this embodiment, the controller object does not cause the display object to
change; the controller object may modify the data, which may cause a display
object to fire or change. However, there is in this embodiment not a tight
coupling of controller objects (interactions) and display objects (responses).

A display object, besides display, is responsible for four different sets
of information. First, the display object has an update method, that is called on
when an action event triggers an action to update the display. In one
embodiment, this is a response that the view (the view is the drawn object -- for
example, the view of a control is the finished result after the control has been
drawn on a monitor of a computer) needs to be updated in some manner. The
first parameter passed to this method can be a reference to the controller object
that is controlling the state of the object. The second parameter can be a generic
object; in the case where updating the display is necessary, this parameter is a
reference to a graphics object.

Second, the display object has a get margins method, that returns the
value of its margin. An example is a border display object for a button control.
It draws a 2x2 border around the button to give it a three-dimensional look.
When the display object is asked for its margins, this method returns a value of
two for the left, top, right and bottom.

Third, the display object has a get desired size method, that returns a
rectangle representing how much size the object needs to display its entire
contents. An example is a multi-line text display object. Assume the first line
is ten characters long (or, 100 pixels), and 20 pixels high. The second line is 80
character (or, 800 pixels) and 20 pixels high. The displayer object calculates the
size of the longest line, adds a margin between lines (such as five pixels), and
returns a rectangle with a width of 800 pixels and a height of 45 pixels.

Fourth, the displayer object has a get display area method, that returns a
rectangle of how much actual space it is occupying. An example may be a check
box control, a diagram of such a control being shown in FIG. 4. When the
border displayer draws the area around the check mark, it is only drawing in a
small portion of the entire rectangle allocated for the object. This dimension can
then be used by the control to layout and offset the values when it makes calls to
other displayer objects needed to draw the object.

As has been described, a given container object may have more than one
displayer object to actually compose the visual representation of the data within
the data object. This is called chaining or grouping. A displayer group
maintains a list of displayer objects associated with a control. An application
programming interface (API) for a displayer group provides for adding,
removing, inserting and retrieving displayer objects. In one embodiment, there
is no implied policy in the order in which drawing is to take place; rather, the
drawing order is determined by the implementation of the objects.

Thus, in one embodiment the displayer group API has five associated
methods. First, an add displayer method to add a new displayer object to the list.
Second, a remove displayer method to removed a specified displayer object
from the list, which is then compacted to remove the open gap. Third, a remove
displayer at method to remove the displayer at a specified location, the list then
being compacted to remove the open gap. Fourth, an insert displayer method to
insert a specified displayer object at a given location, such that the other
displayer objects are pushed down in the list. Finally, fifth, a get displayers
method to retrieve the list of displayer objects contained in the group.

As has also been described, displayer objects may have responses that are
non-visual in nature. Thus, even though a displayer object receives an update
notification when it is supposed to update its view, the displayer object does not
actually have to draw. An example is a displayer that plays a sound when the
pointer enters its area and a different sound when it leaves. The sound controller
instantiates the appropriate displayer based on the interpretation of the raw input,
and the displayer is still called via the update method. However, instead of
passing the graphic object, a reference to a sound object is instead passed, and
the displayer object uses this object to actually play the sound.

In one embodiment of the invention, as can be appreciated by those of
ordinary skill within the art, the displayer object and the displayer group have
interface specified in C++ as follows:

```java
public interface IDisplayer
{
    public void onUpdate(IController controller, Object object);
    public size getDesiredSize();
    public Rect getDisplayArea();
    public Rect getMargins();
}
```

```java
public interface IdisplayerGroup extends Idisplayer
{
    public void addDisplayer (Idisplayer displayer);
    public void removeDisplayer (Idisplayer displayer);
    public void removeDisplayerat (int position);
    public void insertDisplayer (Idisplayer displayer, int position);
    public Idisplayer[] getDisplayers();
}
```

**Conclusion**

An object model has been described. Although specific embodiments
have been illustrated and described herein, it will be appreciated by those of
ordinary skill in the art that any arrangement which is calculated to achieve the
same purpose may be substituted for the specific embodiments shown. This
application is intended to cover any adaptations or variations of the present
invention. Therefore, it is manifestly intended that this invention be limited only
by the following claims and equivalents thereof.
We claim:

1. A system comprising:
   a plurality of container objects, each container object comprising:
   a data object to store data;
   a controller object to provide at least one interaction with the data
   of the data object; and,
   a display object to provide at least one response for the data of
   the data object via the controller object.

2. The system of claim 1, wherein the data of the data object of each
   container object is organized via a plurality of static properties.

3. The system of claim 1, wherein each container object further comprises a
   data cache object to temporarily store data organized via at least one dynamic
   property for the data object of the container object.

4. The system of claim 3, wherein the data cache object of each container
   object directly governs all access to the data object of the container object.

5. The system of claim 1, wherein the controller object of each container
   object at least indirectly governs all access to the data object of the container
   object.

6. The system of claim 1, wherein the interactions of the controller object of
   each container object comprise at least one of messages and events.

7. The system of claim 1, wherein the responses of the display object of
   each container object comprise at least one visual representation of the data of
   the data object.
8. The system of claim 1, wherein the response of the display object of each container object comprise at least one non-visual representation of the data of the data object.

9. The system of claim 1, further comprising at least one external object, each external object accessing the data object of a container object via the controller object of the container object.

10. A container object comprising:
    a data object to store data organized via a plurality of static properties;
    a controller object to provide at least one interaction with the data of the data object, each interaction comprising one of a message and an event; and,
    a display object to provide at least one response for the data of the data object via the controller object, each response comprising one of a visual representation of the data and a non-visual representation of the data.

11. The container object of claim 10, further comprising a data cache object to temporarily store data organized via at least one dynamic property for the data object.

12. A computer-readable storage medium having stored thereon data representing a container object comprising:
    a data object to store data organized via a plurality of static properties;
    a controller object to provide at least one interaction with the data of the data object, each interaction comprising one of a message and an event; and,
    a display object to provide at least one response for the data of the data object via the controller object, each response comprising one of a visual representation of the data and a non-visual representation of the data.

13. The computer-readable storage medium of claim 12, wherein the
container object further comprises a data cache object to temporarily store data organized via at least one dynamic property for the data object.

14. A computer comprising:

5 a processor;

a memory; and,

data residing in the memory for execution by the processor and representing:

a data object to store data organized via a plurality of static properties;

a controller object to provide at least one interaction with the data of the data object, each interaction comprising one of a message and an event; and,

a display object to provide at least one response for the data of the data object via the controller object, each response comprising one of a visual representation of the data and a non-visual representation of the data.

15. The computer of claim 14, wherein the container object further comprises a data cache object to temporarily store data organized via at least one dynamic property for the data object.
FIG. 2
INTERNATIONAL SEARCH REPORT

A. CLASSIFICATION OF SUBJECT MATTER
IPC 7 G06F9/44

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED
Minimum documentation searched (classification system followed by classification symbols)

IPC 7 G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic database consulted during the international search (name of database and, where practical, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

<table>
<thead>
<tr>
<th>Category</th>
<th>Citation of document, with indication, where appropriate, of the relevant passages</th>
<th>Relevant to claim No.</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>LIN L-H ET AL: &quot;DYNAMIC WINDOW CONFIGURATION IN AN OBJECT ORIENTED PROGRAMMING ENVIRONMENT&quot; PROCEEDINGS OF THE ANNUAL INTERNATIONAL COMPUTER SOFTWARE AND APPLICATIONS CONFERENCE. (COMPSAC), US, WASHINGTON, IEEE COMP. SOC. PRESS, vol. CONF. 13, 1989, pages 381-388, XP000091530 page 382, paragraph III</td>
<td>1, 5, 7, 9, 10, 12, 14</td>
</tr>
</tbody>
</table>

Further documents are listed in the continuation of box C.

Patent family members are listed in annex.

* Special categories of cited documents:

'A' document defining the general state of the art which is not considered to be of particular relevance

'E' earlier document but published on or after the international filing date

'L' document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

'O' document referring to an oral disclosure, use, exhibition or other means

'P' document published prior to the international filing date but later than the priority date claimed

'T' later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

'X' document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

'Y' document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

'S' document member of the same patent family

Date of the actual completion of the international search: 14 March 2000

Date of mailing of the international search report: 22/03/2000

Name and mailing address of the ISA
European Patent Office, P.B. 5618 Patentlaan 2
NL - 2280 HV Rijswijk
Tel. (+31-70) 340-2040, Tx. 31 651 epo nl
Fax: (+31-70) 340-3016

Authorized officer: Bijn, K
### INTERNATIONAL SEARCH REPORT

**C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT**

<table>
<thead>
<tr>
<th>Category</th>
<th>Citation of document, with indication, where appropriate, of the relevant passages</th>
<th>Relevant to claim No.</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>JAAKSI A: &quot;IMPLEMENTING INTERACTIVE APPLICATIONS IN C++&quot; SOFTWARE PRACTICE &amp; EXPERIENCE, GB, JOHN WILEY &amp; SONS LTD. CHICHESTER, vol. 25, no. 3, 1 March 1995 (1995-03-01), pages 271-289, XP000655503 ISSN: 0038-0644 page 272, line 14 -page 281, line 2</td>
<td>1,2,5-7, 10,12,14</td>
</tr>
<tr>
<td>X</td>
<td>&quot;HETEROGENEOUS OBJECT MODEL-VIEW RELATIONSHIPS&quot; IBM TECHNICAL DISCLOSURE BULLETIN, US, IBM CORP. NEW YORK, vol. 33, no. 6B, 1 November 1990 (1990-11-01), pages 242-243, XP000108853 ISSN: 0018-8689</td>
<td>1,5-9</td>
</tr>
<tr>
<td>A</td>
<td>the whole document</td>
<td>10-15</td>
</tr>
<tr>
<td>A</td>
<td>EP 0 757 313 A (INTERGRAPH CORP) 5 February 1997 (1997-02-05) page 5, line 50 -page 7, line 59</td>
<td>1,3, 10-15</td>
</tr>
<tr>
<td>A</td>
<td>US 5 608 909 A (ATKINSON ROBERT G ET AL) 4 March 1997 (1997-03-04) column 6, line 1 - last line</td>
<td>1,3,4, 10-15</td>
</tr>
<tr>
<td>P,X</td>
<td>US 5 892 949 A (NOBLE ALAN C) 6 April 1999 (1999-04-06) column 5, line 39 -column 8, line 11</td>
<td>1,2,5-7, 10,12,14</td>
</tr>
<tr>
<td>Patent document cited in search report</td>
<td>Publication date</td>
<td>Patent family member(s)</td>
</tr>
<tr>
<td>---------------------------------------</td>
<td>-----------------</td>
<td>-------------------------</td>
</tr>
<tr>
<td>US 5892949 A</td>
<td>06-04-1999</td>
<td>NONE</td>
</tr>
</tbody>
</table>