

object's properties should remain unchanged. If the apply OPL function has been invoked in standard mode, the apply OPL function determines all of the object's properties that were not specified in the input OPL (step 910). Next, the apply OPL function changes these unspecified properties to their standard value (step 912). In this step, the apply OPL function accesses the mapping of properties to their standard values and sets the unspecified properties of the object to their standard value.

Applications

Following is a description of four applications that take advantage of the benefits of the OPL of the preferred embodiment and which use the previously described functions. The first application is the arithmetic application which has been previously described with reference to FIGS. 1A-1F. In the arithmetic application, a user has modified an original object and the modifications are then isolated and applied to another object. FIG. 10 depicts a flowchart of the steps performed by the arithmetic application. The first step performed by the arithmetic application is to invoke the generate OPL function on the modified object (step 1002). This step will return an OPL describing the properties of the modified object. The generate OPL function is then invoked on the original object (step 1004). This step will return an OPL describing the properties of the original object. Next, the subtract OPL function is invoked with the OPLs returned from both step 1002 and 1004 being passed as parameters (step 1006). In this step, the subtract OPL function will isolate the modifications and will return an OPL containing the modifications. After isolating these modifications, the apply OPL function is invoked on a given object with the OPL returned in step 1006 as a parameter (step 1008). After the processing of this step, the object to which the modifications have been applied will have its properties modified accordingly.

FIG. 11 depicts a flowchart of the steps performed by a format painting application. The format painting application provides for the property settings (or format) of a particular object (source object) to be extracted from the source object and applied to another object (destination object). For example, all of the property settings for object 100 in FIG. 1B can be obtained and applied to an object of a different type, such as object 110 in FIG. 1C. This functionality will have the effect of changing all of the properties in object 110 to the settings of the object 100 if the properties are common among the two objects. In other words, if a property is specified in the property list obtained from object 100 which object 110 does not have, this property is ignored. The first step performed by the format painting application is to generate an OPL from the source object (step 1102). In this step, an OPL is generated in Ninch mode so that a property that the source object does not have is not reset on the destination object. That is, if a standard mode OPL were used for the source object and a value were unspecified because the property was unknown, it would be set to its standard value in the destination object even though this action was unintended. Next, the format painting application invokes the apply OPL function and applies the OPL returned from step 1102 to the destination object (step 1104). In this step, the Ninch mode OPL returned from step 1102 is applied to the destination object and the destination object will henceforth adopt the property values of the source object. Again, additional or different properties that the destination object has that the source object does not are unaffected by the format painting application.

FIG. 12 depicts a flowchart of the steps performed by a synchronization application. The synchronization applica-

tion synchronizes the properties of one object (first object) with the properties of another object (second object). As such, when the properties of the first object are modified, the synchronization application automatically makes the same modifications to the corresponding properties (i.e., properties having the same property ids) of the second object. The first step performed by the synchronization application is to determine if a property of the first object has been modified (step 1202). This step can be performed by polling the first object at predetermined intervals. When polling the object, the property list of the object is obtained using the generate OPL function and the property list is compared to its most recent OPL, the most recent OPL being retained after each invocation of step 1202. If it is determined that a property has been modified, the synchronization application isolates the modification that was made to the first object by invoking the subtract OPLs function and passing both the OPL of the first object as modified and the most recent OPL of the first object (step 1204). After isolating the modification, the synchronization application applies the modification to the second object (step 1206). The OPL is applied to the second object by utilizing the apply OPL function, after which the synchronization between the properties of the first object with the properties of the second object is achieved.

FIG. 13 depicts a flow chart of the steps performed by a combination application. The combination application provides the user with the ability to combine modifications that were made to two objects (source objects). After combining the modifications, the combined modifications can then be applied to a destination object. This functionality allows a user to pick and choose object formats that they like, combine them, and apply these formats easily to a destination object. The first step performed by the combination application is to isolate the first set of modifications (step 1302). In this step, the modifications made to the first source object are isolated by subtracting the OPL of the source object as modified from the OPL of the source object in its original form. This step is performed by invoking the subtract OPLs function. Next, the combination application isolates the modifications made to the second source object (step 1304). The processing of this step is similar to that as described relative to step 1302. After isolating both the modifications to the first source object and the modifications to the second source object, the combination application adds the modifications together (step 1306). This step is performed by invoking the add OPLs function and designating one of the source objects as the overriding source object. After adding the OPLs together, the combination application applies the resulting OPL to the destination object as specified by a user or program (step 1308). After performing this step, the modifications made to the two source objects have been easily isolated, combined and applied to the destination object.

While the present invention has been described with reference to a preferred embodiment thereof, those skilled in the art will know of various changes in form that may be made without departing from the spirit and scope of the claimed invention as defined in the appended claims.

I claim:

1. A computer-readable medium whose contents cause a computer system having a memory containing an object with properties and a computer program for accessing the properties of the object to perform processing, by performing the steps of:

storing a property list containing the properties in a contiguous block of the memory by the object, wherein at least one of the properties is a nested property list; and