

be described below. An object which depends from a warp object is herein referred to as a warped object.

[0083] A warp object may be similar in nature to the Value object described above, as shown below. Again, in order to provide the additional functionality required, the WarpObject provides additional data repositories for storing the additional information.

WARPOBJECT CLASS - TABULATED VIEW		
CLASS NAME: WarpObject		
ATTRIBUTES	Type	Comments
Name	created depending on type specified	
warp_master_value	string	
warp_rule	string	

[0084] The WarpObject class, from which the WarpObject is created, has the necessary accessor methods for providing the required functionality, for example, to obtain the reference of an object from a string containing a relative location of the object (using, for example, the above-described tree navigation functionality). Similarly, the warp object accessor methods also provide the required functionality to create an object from a class depending on, for example, the value of an attribute as defined by the warp master value and the warp rule.

[0085] The class definition for the Video Card class may thus be expressed as:

VIDEO CARD CLASS - TABULATED VIEW		
CLASS NAME: VideoCard		
ATTRIBUTES:	Type:	Comments
video_memory	enum	Choice: '1 MB', '2 MB', '4 MB'
pixel clock	integer	
display	warp_object:	warp_master_value: ..\..\computer_type; warp_rule: use class Monitor if warp_master_value = 'desktop'; use class LCD_screen if warp_master_value = 'laptop'

[0086] The pseudo code for the VideoCard class declaration may thus be expressed as:

```

VIDEOCARD CLASS - PSEUDO-CODE

class VideoCard; # define the video card class
{
  attributes: # define the attributes required by the VideoCard class
  {
    video_memory: # define the characteristics of the variable
    # video_memory
    {
      type : enum ; # it's of type enum
      # having a choice as defined...
      choice : ('1MB', '2MB', '4MB' );
    }
    pixel_clock:
  }
}
    
```

-continued

```

VIDEOCARD CLASS - PSEUDO-CODE

{
  type: integer;
}
display:
{
  type: warp_object; # declares that the monitor is of type
  # warp_object class
  warp_master_value: ..\..\computer_type;
  warp_rule:
  {
    use class Monitor if warp_master_value =
    'desktop';
    use class LCD_screen if warp_master_value =
    'laptop';
  }
}
}
    
```

[0087] The display attribute declaration is used to create a warp object depending from the Video Card object. The warp object uses the warp master value (i.e. the computer type attribute 103) and the warp rules to determine which object (in this case a Monitor or LCD screen object) should be created depending on the value of the warp master, as illustrated in FIG. 4. Through the warp object a call, for example, to set the screen resolution, will be directed to either the screen resolution attribute 110 of the monitor object or to the screen resolution attribute 120 of the LCD screen object 116, depending on which object has been created.

[0088] FIG. 7 is a flow diagram outlining one way in which the main steps may be performed according to an embodiment, using a Ped-type implementation, for requesting the value of screen resolution (step 700). If the warp object (105) has not been created yet (step 702) it is created (step 704). The warp master value is obtained (step 706) and, for example using the previously mentioned tree navigation functionality, the reference of the warp object is registered with the warp master value (step 708). Providing that the warp master value has been defined (step 710) the warp rule is obtained (step 712) and is applied to the warp object (step 714) leading to the creation of the warped object (step 720). In the present example either a Monitor object or an LCD screen object will be created, as defined by the warp rule in accordance with the value of the computer type attribute 103. The value of the screen resolution attribute of the created warped class is then returned to the requesting function.

[0089] If the computer type attribute 103 is changed, for example from 'desktop' to 'laptop', then the warp object 105 will, using the above-described registration mechanism, create an LCD screen object 116, and all future accesses will be redirected to this object. Preferably the attribute values of the monitor object will be copied over to the LCD screen object where possible, for example using a 'best effort' copy. Any attribute values which do not exist in the LCD screen object will therefore not be copied across. Preferably the warp object 105 performs this functionality.

[0090] A third type of dependency which may exist between classes is where an attribute value of a class is