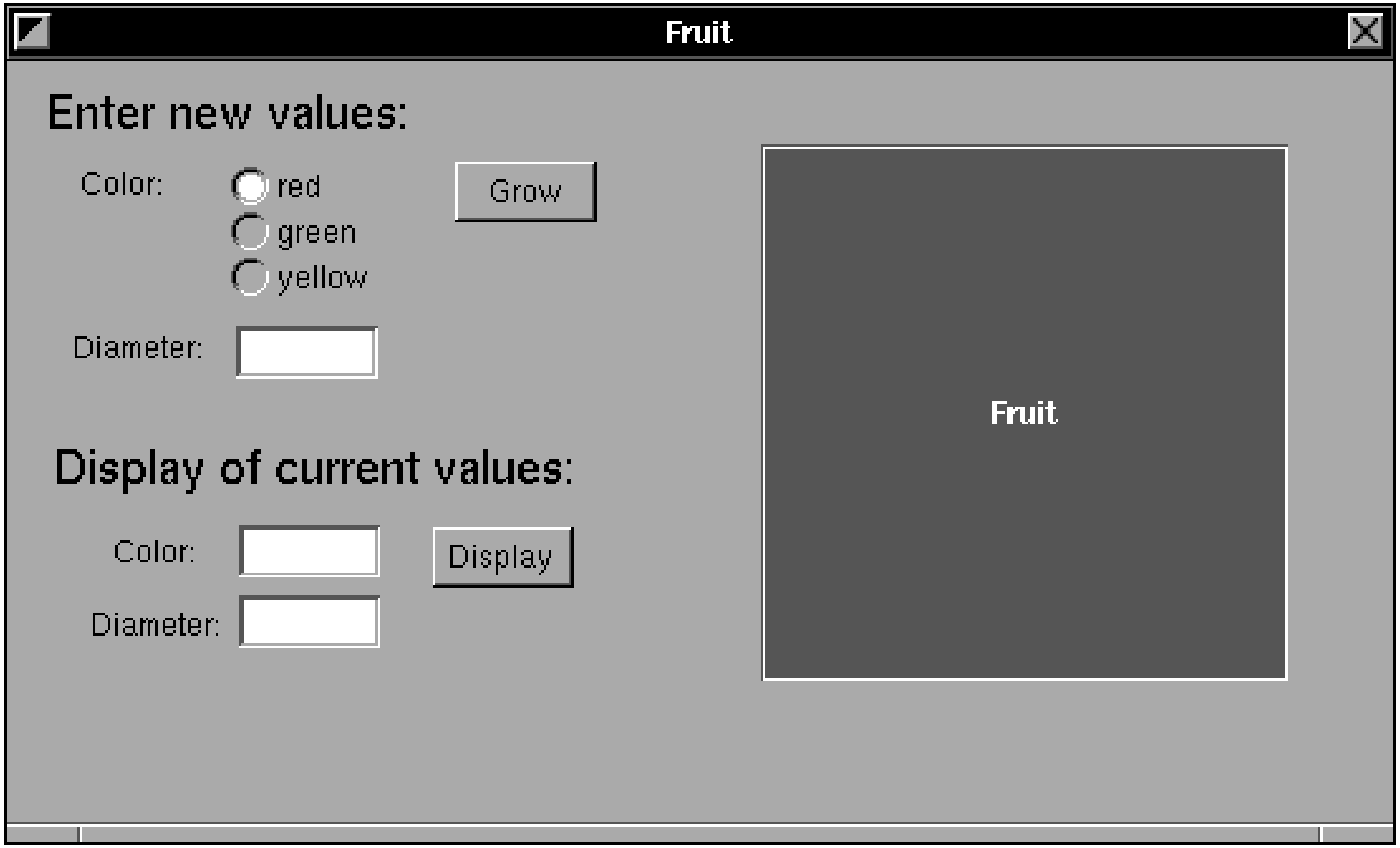
OOP Class
FruitView

Purpose: Using a CustomView object from an IB palette, you will design your own view for an IB interface version of Fruit. You will also learn how to import class files from some other application into the new application.



- To represent the generic piece of fruit you can use the code from the **drawself::** method in the earlier program that drew a square, but slightly modify it to be a filled square rather than a stroked square, or even modify it to draw a filled circle instead.

- The fruit being drawn should get its size from the **diameter** instance variable such that when the fruit grows, the drawing of the fruit gets larger.

- The fruit should get the gray level of its fill from the **color** instance variable:
  color = "red" --> gray level = 0.0 (black)
  color = "green" --> gray level = 0.4 (a dark gray)
  color = "yellow" --> gray level = 0.7 (a very light gray)

The procedure is:

1. First, create a new project in Project Builder named **FruitView**, open its nib file in IB, and draw the screen interface as shown below. (Or, instead of drawing it brand new, you can copy the one from the **IB_Fruit** lab in the following way:

a. First delete the window provided by IB in the current lab by selecting it and pressing delete. When asked to verify, click "Yes".

b. Double-click on Fruit.nib in the **Solution** directory for the **IB_Fruit** lab. This will load that .nib file into IB.

c. Select the desired window and press **Copy**.

d. Now close the Fruit.nib file (**Document**/**Close,** or click the close button in its File window).

e. Select the File Window in the lower left corner of IB for the current lab.

f. Press **Paste**. This should copy the Fruit.nib window to this .nib file. It is not yet connected to other objects, but you will do that later.)

2. Replace the color input textfield with three radio buttons to choose either red, green, or yellow. To include the third button, Alt-drag on the button matrix. Double-click the first button of the matrix to select it individually. In the inspector, check that its tag is already zero. Now select the second and third buttons, setting their tags to 1 and 2, respectively.

Enlarge the main window so that there is room for the view on the right. Drag a custom view onto the window and make it about three inches square. For the Grow button, set it to be Continuous so that you can make the Fruit grow fast by just holding down the Grow button with the mouse. Save (**Document**/**Save**) the .nib file.

3. Now copy Fruit.m and Fruit.h from the **IB_Fruit** lab in the following way: in PB, select **Classes** in the file browser and double-click the suitcase to bring up an Open panel. In the panel, find and select Fruit.m from the IB_Fruit lab Solution directory and click OK. This will copy Fruit.m and Fruit.h into the project.

4. Use Edit and modify the class files to work with the new interface:

a. First, modify the .h file to indicate that Fruit inherits from View, not Object.

b. Modify the .m file (and .h file) such that the method that takes the color from the interface works with the radio button matrix. It should look at the "selectedTag" of the sender to see which button was clicked. (Check the documentation on the Matrix class.)

c. Since Fruit is now a View, you must also change it to have **initFrame:** instead of init.

5. Add the **drawSelf::** method to Fruit.m and Fruit.h, along with any other code you may need. Be sure that the methods for changing color and size send **[self display]** before they return.

6. Add the **appDidInit:** method to Fruit.m and Fruit.h so that the output displays show correct current values when the application starts up.

7. In IB, **Parse** the Fruit class (in the Operations pull down menu). This informs IB of the outlets and action methods owned by Fruit so that you can make connections.

8. Click on the CustomView and go to the Inspector (Attributes). It should be a CustomView Inspector. Choose Fruit as the new class for the CustomView.

9. Make connections (control-drag) to the outlets and actions of the Fruit view (connect directly to/from the Fruit view instance in the window). Connect the **delegate** outlet of the File's Owner icon in the File Window to the Fruit view instance so that **appDidInit:** will get sent to the Fruit instance at application startup.

10. Save (**Document**/**Save** or **command-s**) the interface file again. Return to PB, save the project, and build and run the application.

Extensions: Try drawing a fancier fruit, perhaps a circle instead of a square. Replace the Color radio button matrix with a ColorWell, and set the color of the fruit from the ColorWell, making other changes accordingly. (Use

NXSetColor() in **drawSelf::** to set the current color for drawing.) Or even go further: allow a color chip from the Color Panel to be dragged and dropped on top of the fruit in order to change the color (View has a method for doing this: **acceptColor:atPoint:**).