onViz QuickStart

This mini-tutorial is designed to quickly get your feet wet with some of the powerful tools and features of onViz. As such, we will introduce you to several basic features, without going into the detail of our in-depth tutorial (*still a work-in-progress at this time*). This tutorial is designed as a 'do this – see this' development and will give you a feel for how you can develop applications using onViz' visual authoring environment.

After going through this mini-tutorial, you can explore our more in-depth tutorial (*not yet available online*), which contains step-by-step instructions and detailed information about the onViz tools, or venture off on your own, using the onViz documentation as reference.

If you're new to multimedia authoring, don't be intimidated. Just take it one step at a time and before you know it, you will be creating your very own interactive presentations!

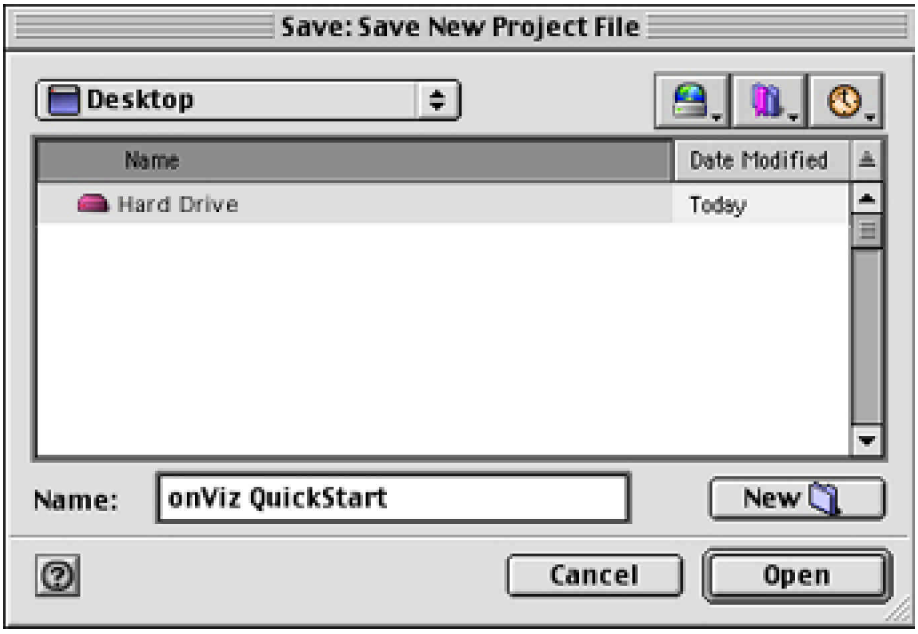
Although it's not absolutely necessary, it is helpful to familiarize yourself with the onViz authoring environment before beginning the mini-tutorial. You can do this by taking the onViz Tour.

When you're ready, open onViz and let's get started! (Note: this tutorial assumes you have basic computer skills)

1) Double-click on the onViz program icon. After the onViz splash screen, you are presented with the onViz Setup Helper with the **New Project** option selected.

The Setup Helper will take you through a series of screens that allow you to set options for your application.

- For now, just click the **Quick Setup**
- Name your application *"onViz QuickStart"*



- Click the **New** button to create a new folder for your application, name the new folder "*My onViz QuickStart*" and then click **Create**.

It is typically a good idea to create a new folder for your onViz projects to keep your project and related support folders organized.

- Click the **Save** button to save your empty onViz project file and begin your new project.

onViz opens to an empty Application Map

The onViz Application Map contains tools in a floating palette that you will use to create your applications.

2)  Click on the Start State icon and drag it to the top center of the Application Map.

Typically, most applications will begin with this state.

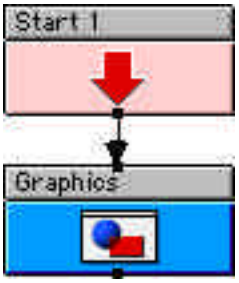
3)  Drag an Output Presentation State below or to the right of the Start State.

You can actually drag it wherever you like, but it is most logical to build your programs from top to bottom or from left to right (or some combination of both).

➢ Rename the Output State from "*Output 1*" to *"Graphics"*.

You can rename any state by clicking once on the upper 1/3 or the state's icon. Once highlighted, you can begin typing after a second or two - when the title area is highlighted with a red border.

4) Click on the small, black dot (called the Exit Node) at the bottom, center of the Start State and drag until your mouse is anywhere over the Output Presentation State (now titled *Graphics*).

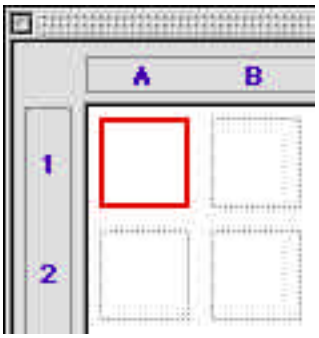


Notice that an arrow (called a Path) automatically connects the exit node of the *Start 1* to the entrance node of *Graphics*. By creating this path, you've just told onViz to move from the Start State to the *Graphics* Output State without having to type any programming code. Cool! (If you need to delete a path, just click on it and hit the "Delete" key.)

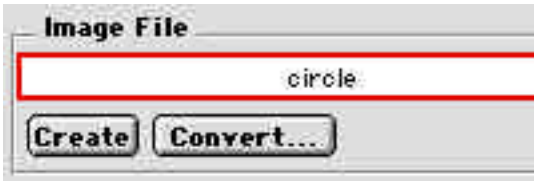
5) Double-click anywhere in the lower two-thirds of the "Graphics" Output State.

This opens the Output State Presentation where you create or place the visual information your audience will see.

The floating palette overlaying the presentation window is the Sprite Attributes Palette (if this palette is not already open, by choosing if from the Window Menu).



➢ Double-click in the first Sprite position (1-A). This opens up the Image Attributes dialog box.



➢ In the Image File section at the top of the Image Attributes dialog, change the name of the graphic from *"Untitled_copy.pct"* to *"Circle"*. Then click the **Create** button.

This takes you to onViz' built-in Image Editor. So, you don't have to have a draw or paint program installed on your computer to create custom graphics.

Of course, you can also use other draw and paint programs to create your graphics and simply import them into onViz.

➢ While in the Image Editor, click on the circle icon in the Object Tool Palette.

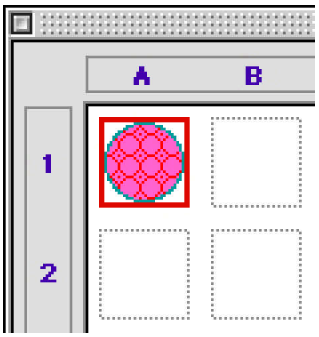
Then click and drag in the Image Editor window to create a circle. (It is assumed that you know how to draw, color and edit basic shapes. onViz' Image Editor will be very easy for you to use if you're familiar with the standard draw and paint programs.)

Feel free to change the fill pattern and color or the pen width and color with the palette at the top of the Image Editor window.

➢ When you have created your circle and have it the way you want, close the Image Editor by clicking in the close box in the upper left-hand corner of the window - or you can close any window by choosing Close Window from the File menu.

➢ Click **Save** when prompted.

Notice that a thumbnail image of your circle is displayed in the first sprite position.



➢ Click and drag the thumbnail onto the presentation window and position it where you like. Then, choose Close Window from the File menu to return to the Application Map.

This is a good time to save your work. As with any computer software, it is a good idea to save often and backup your work. Choose **Save** from the File menu.

Now, let's run your program to see what it looks like so far. You can run your program by holding down the command key (⌘) and typing the letter R. (Throughout the balance of this document, we will simply give the key sequence such as ⌘R.

You will see your circle then immediately a dialog box will pop-up informing you that a "Dead end path was reached". That's because we didn't tell onViz where to go after the application flowed through the *Graphics* Output State.

➢ Click on the **Cancel** or the **Show Me** button to return to the Application Map.

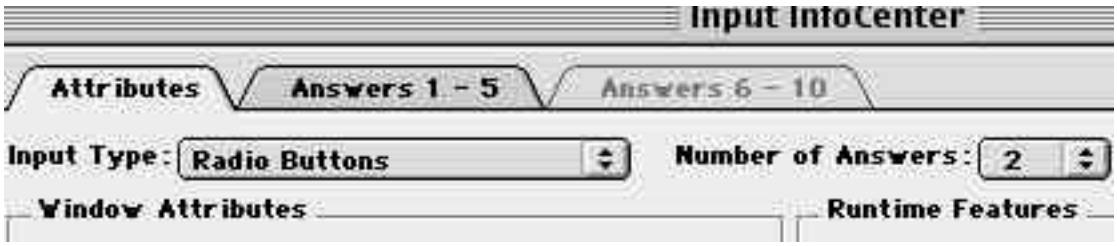
Clicking on the **Show Me** button will highlight the state in which onViz reached the dead end path, which is invaluable when working on large, complex programs.

Keep in mind, onViz executes each state and keeps moving on until it reaches a state where you tell it to stop or pause for some reason – or, as in this case, doesn't know where to go next.

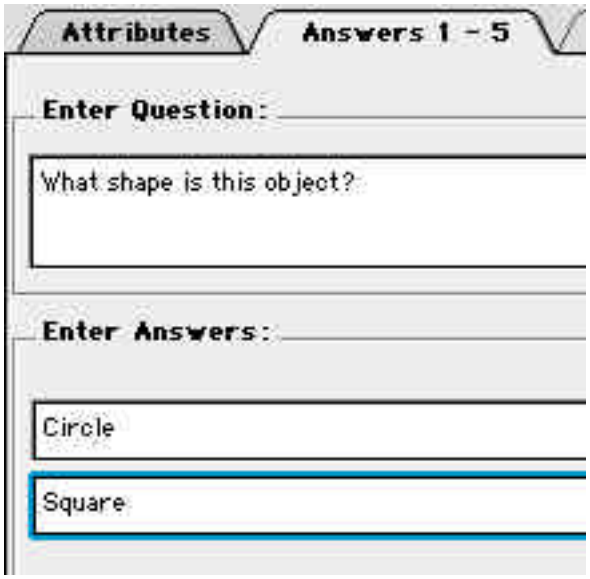
User input can be gathered by using an Input State.

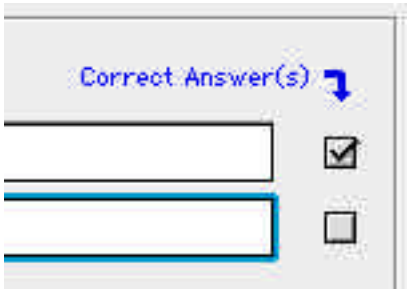
6)  Drag an Input State below the *Graphics* Output State.

- Rename the Input State from *"Input 1"* to *"Question".*
- Drag a path from the exit node of *Graphics* to the input of *Question* so onViz knows to display the circle and then to display a question and wait for the user to submit an answer.



- Double-click the Input State's InfoCenter (top 1/3 of the icon) to open it. While the attributes tab is selected, make sure the **Input Type** is "Radio Buttons" and the **Number of Answers**is "2". You can have up to 10 options, but for this mini tutorial, we'll be using only 2 possible answers.
- Click on the **Answers 1-5** Tab.
- In the Enter Question box, type the question "What shape is this object?"
- In the Enter Answers boxes, type "Circle" for the first answer and "Square" for the second answer.





You may click the Correct Answer(s) box to put a check mark next to Circle, since this is the correct answer, but since we're not tracking scores in this mini tutorial, it is not necessary.

- Click OK to return to the application map and run the program again (⌘ R).

Now you will see the circle along with the dialog box showing your question and the two choices for the answer. Depending on where you positioned the circle image, the dialog box may be covering it up.

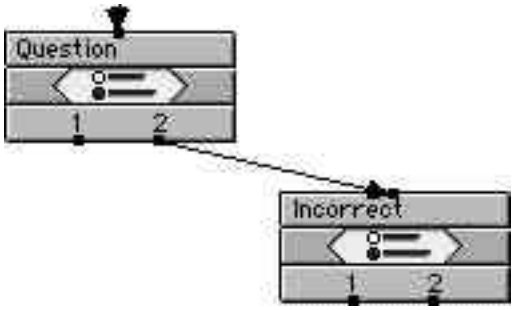
To reposition and/or resize the input dialog box, you need to stop the program (⌘ .) and double-click on the Presentation (bottom 2/3) part of the *Question* Input State.

This opens up the Background Output selection dialog box, where you can select the Output State named *Graphics*. (So far, this is the only Output State in the list, but it's important you name all of your states something meaningful to you so you can find it easily when your file has dozens and dozens of States from which to choose.)

To reposition the box, click on the upper left-hand area. To resize the box, click on the lower right-hand area. When you're done, click on the **OK** button or close the window by typing ⌘ W.

7) Now we will use an Input State to give feedback for an incorrect - in this case, if the user chooses on "Square" instead of "Circle". Drag another Input from the Tool Palette and place it below and to the right of the *Question* Input State. (Again, you can actually place this State anywhere, as long as it's connected to the *Question* Input State properly.)

   - Name this State "*Incorrect*".
   - From the *Question* Input State, connect the Exit Node labeled "2" to the *Incorrect* Input State.

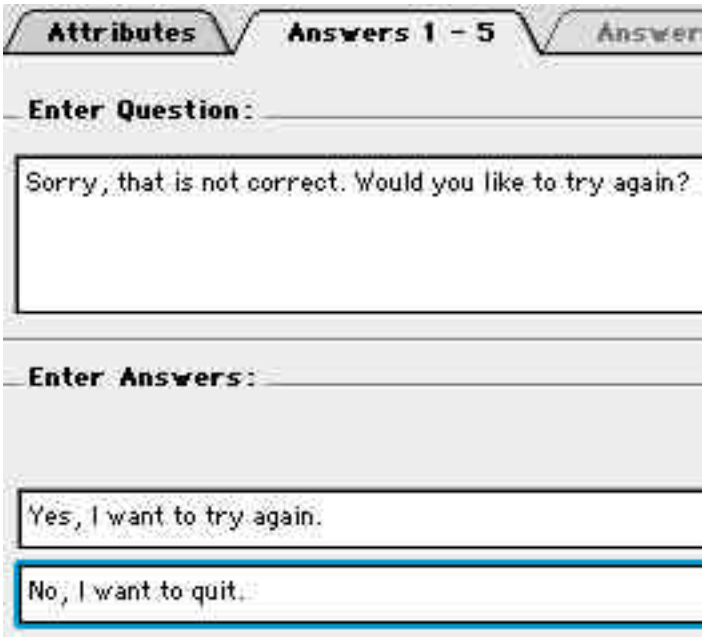


   Remember that we typed in "Square" as the second of the two possible answers. (You can confirm this by clicking on the Exit Node itself and reading "Square" in the bottom 1/3 of the *Question* Input State.)

   - Now let's go to the InfoCenter of the *Incorrect* Input State.

   In this instance, we are going to be using the Input State to provide feedback. If we were going to be tracking scores in our application, we would probably deselect the **Report Results** option.

- Make sure the **Input Type** is “Radio Buttons” and the **Number of Answers** is “2” (as in step #9), because we’re going to give the user the option to try again or exit after reading the feedback.
- In the Answers 1-5 tab click in the **Enter Question** field and type “Sorry, that’s not correct. Would you like to try again?”
- In the **Enter Answesr** fields, type “Yes, I want to try again.” in the first field and “No, I want to quit.” in the second field.



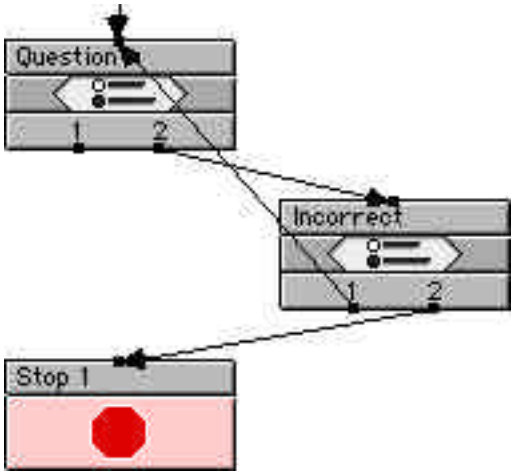
You can resize your Input window now by clicking the **Attributes** tab and then clicking the **Position…** button in the Window Attributes section of the dialog.

This opens the Background Output selection dialog box so you can resize and position your Input window where you want it to display.

- Click on “OK” to return to the Application Map.

8)  Drag a Stop State below and to the left of the *Incorrect* Input State.

- Connect the second Exit Node of the *Incorrect* Input State to the Stop State. Remember, the second choice we created says, “No, I want to quit.” You’ve just told onViz to exit the program after the user chooses this option.
- Connect the first Exit Node of the *Incorrect* Input State to the Entrance Node of the *Question* Input State. By doing this, you’re telling onViz to display the question again when the user chooses the first answer – “Yes, I want to try again.”



This is a good time to save your work (⌘ S).

Now, what if the user chooses the correct answer? Well, we could use an Input State for feedback just as we did for the incorrect answer. Or, perhaps use another Output to give more a more graphical feedback response.

However, this is a good opportunity to introduce the Calculator State. (Once you get a grasp on what the Calculator State can do, you can move beyond creating simple slide show presentations and really start creating complex, interactive multimedia programs!)

Before we work with the Calculator State, we need to create the feedback the user will see when the correct answer is chosen.

9) To do this, let's double-click on the *Graphics* Input State Presentation Window (the lower 2/3rds of the state icon).

Make sure the Sprite Attributes Palette is open. If the palette is not visible, you can open it from the Window Menu on the top toolbar.

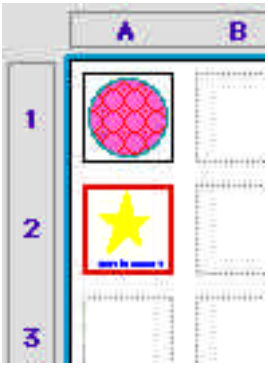
➢ Double-click in the second sprite position, 2-A.

This will take you to the Image Attributes Window.

➢ Name the graphic "Correct Answer Feedback" and then click on the "Create" button.

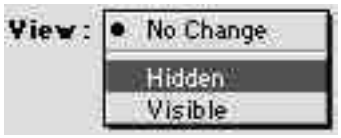
This will take you to the image editor where you will design the graphic and/or text that the user will see when the correct answer is chosen.

For example, you might use the polygon tool to draw a star along with the text, "That is correct!" below it.



➢ When you're done drawing your picture, close the Image Editor, drag the thumbnail in 2-A to the Presentation Window and place it where you would like it displayed when the user gives a correct answer.

➢ In the **Reset Sprite Variables** section of the Sprite Attributes Palette (lower right), click on the **View** dropdown menu and select **Make Hidden** (make sure Sprite 2-A is selected).



When running the application, this option causes any graphics/text in sprite position 2-A to be invisible until onViz tells it to display it later (after the user selects the correct answer.).

➢ Now close the window (⌘ W) and return to the application map.

Now we need to tell onViz to show the correct feedback when the correct answer is selected.

10)  Drag a Calculator State below and to the left of the *Question* Input State.

- Name the calculator “Correct”.
- Connect the left Exit Node of the *Question* Input State to the Entrance Node of the *Correct* Calculator State. This tells onViz to take that path when the user selects the correct answer– in this case “Circle”.
- Connect the Exit Node of the Calculator State to the Stop State so onViz knows to end the program when it is done executing the Calculator State, which we will set-up in the next step.
- Open the Calculator State by double-clicking anywhere on the State.
- Click on the Question Mark (**?**) button.

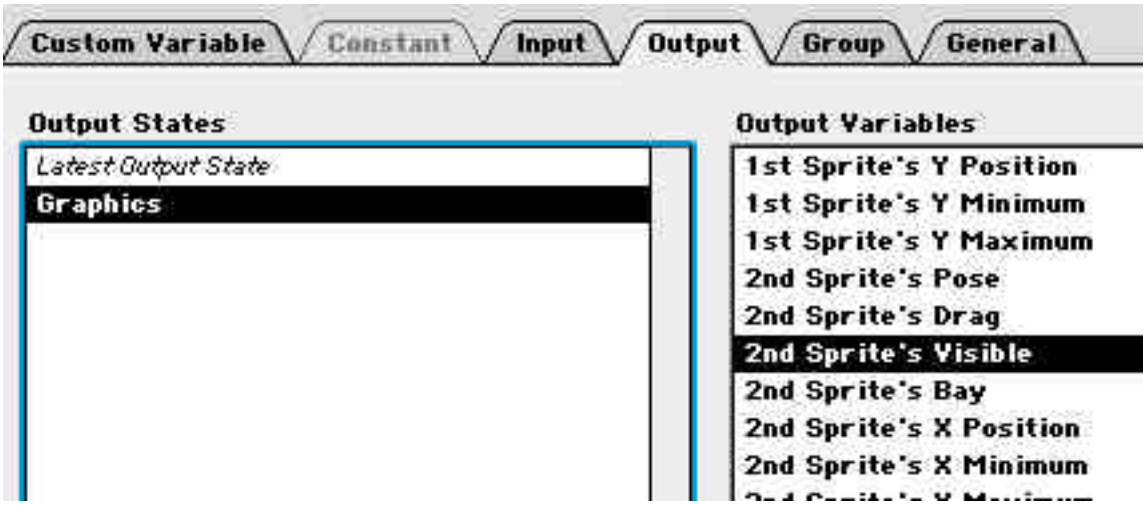
This displays the Variable Selection dialog allowing you to choose from dozens of built-in onViz variables.

- Click on the **Output** tab.

This shows a list of variables that affect Output States. In this case, we have used only one Output State (named *Graphics*) which you should see in the list of Output States.

- Click on*Graphics* in the list of Output States to select it. In the list of Output Variables, click on “2nd Sprite’s Visible”.

You select this Output Variable because we want to change the visibility of Sprite number 2 (your ‘Correct’ feedback) when the application flows through this calculator.



- Click the **Select** button to close the Variable Selection dialog.

Now, instead of a question mark, you can see the variable you just selected on the left side of the variable equation.

- Click on the right side of the equation and type the number “1” (which means “Yes” or “True” to onViz. “0” means “No” or “False” to onViz.)

So, to put it in plain English, you’ve just told onViz to make the graphics contained in the 2nd Sprite of the Input State called *Graphics* visible.

- Click on the “Immediate Update” checkbox on the right side of the Calculator Window .

Selection this option causes onViz to make the sprite visible as soon as this equation is executed. Without this option selected, onViz won’t make the sprite visible until it re-enters this State, which won’t happen in this case since after the calculator, the application flows to the Stop State.

- Click **OK** to return to the Application Map.

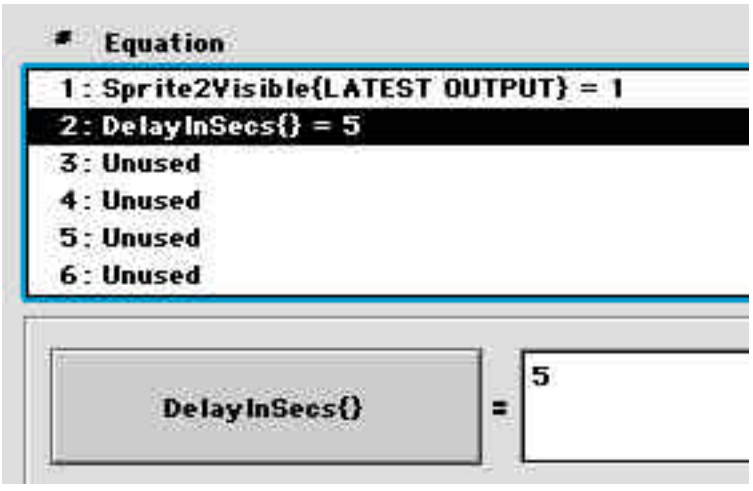
Now, let’s run (⌘ R) the program again. First, click the incorrect answer to see your feedback dialog.

Click the try again option to see your question repeated and click the correct answer. You’re graphic shows for an instant and then the program ends.

We need to pause onViz long enough for the user to read the feedback. We can do this within the same *Correct* Calculator State.

11)Open the *Correct* Calculator State.

- Click on the 2nd Equation.
- Click on the Question Mark (**?**) button on the left side of the equation then click the **General** tab.
- Click on the Delay in Seconds variable to select it, then click **Select**.
- Click on the right side of the equation and type “5”.



onViz sees that “5” as five seconds. It’s important that this variable is executed *after* the equation that sets the sprite visibility. Otherwise, onViz would pause for 5 seconds then show your feedback for an instant before exiting.

By positioning the variables in the order they are, onViz will your feedback visible, then it will pause for 5 seconds before exiting.

➢ Click “OK” and run your program now and checkout the results!

Once you have your application the way you want in the *authoring* environment, now you can build it as a standalone application so you can see how you users will see it in the *runtime* environment.

12) Choose the **Editor** from the *File > Build* menu.

The Build Editor gives options for your standalone application – the target platform, whether the application will be delivered in a networked environment, bookmark and report storage location, etc.

To test your new application, we will will use the default settings, a single file application with the player included in the Top Level.

If you want to build both Macintosh and Windows applications simultaneously, rename one of the default ‘builds’ accordingly and select both **Macintosh** and **Windows** options.

➢ Click **Build** in the lower right of the Build Editor dialog.

➢ Click **Save and Build** on the verification dialog that is display.

onViz automatically builds standalone application for both Macintosh and Windows and saves them in a folder named “*Build Targets”* in the same folder as your onViz project.

You can now double-click your “*onViz QuickStart”* application and see how your user will see the application.

Congratulations!

You've just created a multimedia program that goes beyond a standard presentation. After you play around with onViz and explore it's many other features, you will be making dynamic, smart programs that are truly interactive.

Watch the onViz message board and file download site for the latest information about onViz!

For further information or technical support, contact:

onViz Development Team
Discovery Systems International, Inc.
PO Box 1188
Knoxville, TN 37901-1188
Phone: 888.284.5389
Fax: 865.609.2460
e-mail: onviz@discoverysystems.com
www.discoverysystems.com/onviz/onviz.htm