



VisualEyes Tutorial

About this tutorial

This tutorial will introduce you to working with VisualEyes by walking you through the creation of a project: "After the Greenwich Tea Party: New Jersey During the American Revolution." In [Chapter One](#), you will create a new project designed to plot the battles and skirmishes of the Revolutionary War in New Jersey. In [Chapter Two](#), you will learn how to edit your project and add features. In [Chapter Three](#), you will learn how to create paths and dots. In [Chapter Four](#), you will learn how to create widgets and import data into your project using external files. And in [Chapter Five](#), you will learn how to present the data in infoBoxes (short for "information boxes") that can interact with the viewer.

Chapter One

In this chapter, you will learn the basics of VisualEyes:

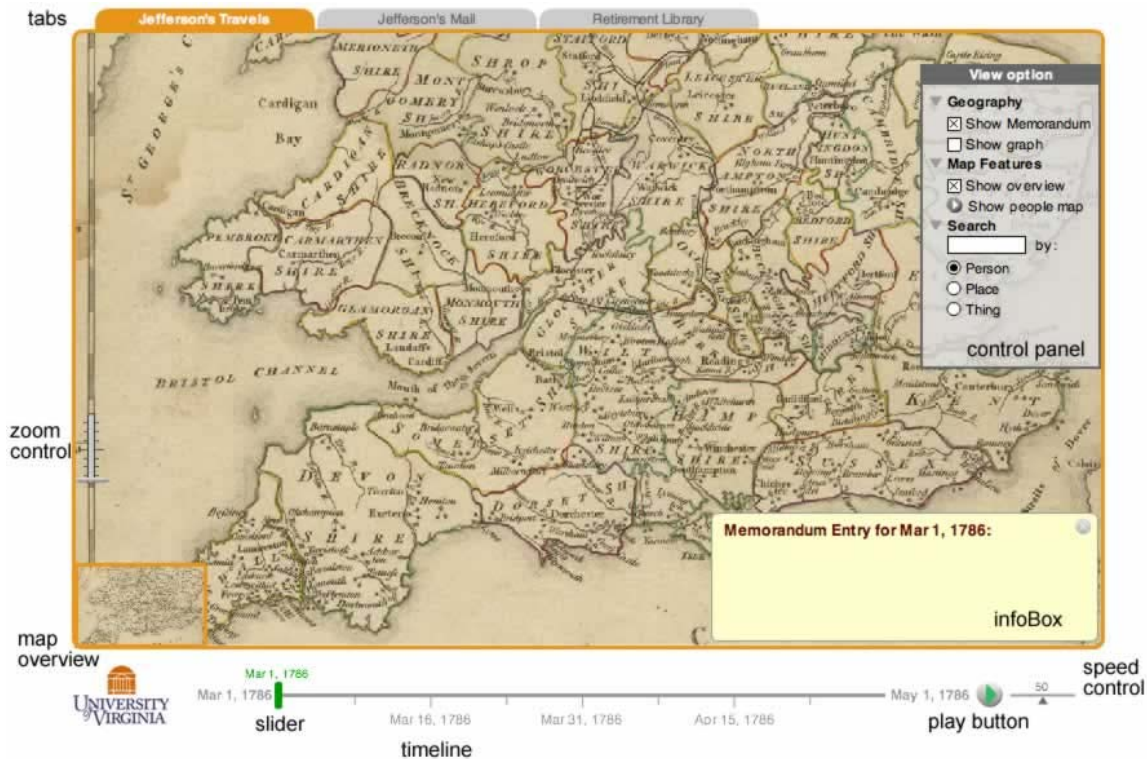
- Projects, Views, Resources, Displays, and Controls
- Introduction to the VisEdit Tool
- You will learn how to:
 1. Create a new project
 2. Understand VisEdit
 3. Understand elements and sub-elements
 4. Edit a project with VisEdit
 5. Save a project with a new name

Understanding VisualEyes: Projects, Views, Resources, Displays, and Controls

Before you begin creating a new VisualEyes project, you'll find it helpful to take a look at an existing one.


- Point your browser to <http://www.viseyes.org/show/?base=jt> to view the Jefferson's Travels visualization project.

- Take a few minutes to examine the screen.



The entire set of features you see on the screen – the set of three tabs, the timeline, any additional instructions, text and images – make up the *project*. Each tab represents a *view*, a set of features designed to present one aspect of the material. The information that goes into the view – the maps, text, images, and data – are called the *resources*. And the building blocks that allow you to interact with the resources – such as the *map overview*, *timeline*, and *control panel* – are called the *displays* and *controls*.


The first tab, Jefferson's Travels, gives a geospatial representation of Jefferson's travels in England in 1786. The background map shows southern England in the 18th century. A *slider* on the left side allows you to zoom in and out. When you zoom in, the *map overview*, in yellow, allows you to keep track of where you are on the large map.

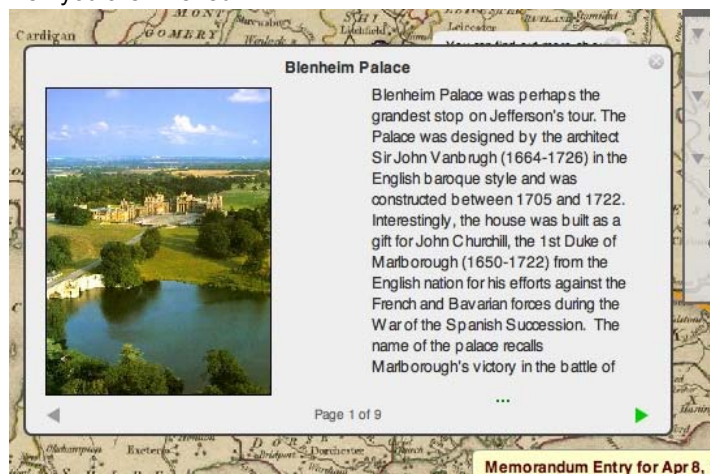
To track Jefferson's travels across the map, you can press the Play button  on the *timeline* at the bottom of the project. Beside the timeline is a slider that lets you control the speed of the timeline; slow it down or speed it up to suite your needs.


- Click on the Play button to watch the path of Jefferson's travels. A small portrait of Jefferson traces his **path** across the map. That **path** is linked to the **timeline**, so that Jefferson's travels can be pinpointed very precisely in time as well as space. Both **path** and **timeline** are linked to an **infoBox** showing Jefferson's memoranda for each date.

Note that the map switches at two points, to a map of 18th century London. This allows viewers to chart Jefferson's travels more closely.

Explore other features of the Jefferson's Travels view:

- Stop the movie by pressing the *pause button*.
- Use the timeline slider to position the timeline on April 8, 1776
- Click on an *information button* . A **docviewer** appears, providing images and text about the location. Click the arrow buttons to move through the pages, and the close button when you are finished.



- Click on a building *icon* . This brings up an **infoBox** that links to an external web page. Click the close button when you are done.

Next, take a look at the Control Panel, on the right of the view. This allows the user to display, or hide, other resources included in the view.

- Look under Geography. When you first load the Jefferson's Travels view, the "Show Memorandum" box is checked. Click on it to remove the check, and to hide Jefferson's memoranda.
- Click on "Show Graph" to display a graph of Jefferson's purchases, and on "Show Overview" under Map Features to remove the map overview.

The next feature on the Control Panel, "Show People Map," demonstrates a valuable feature of VisualEyes, a *concept map*. It allows viewers to explore networks of interrelated people, places, or things.

- Click on "Show People Map" to display Jefferson's social network while he was in England.
- At the center is Jefferson himself. Click on the image to see an *infoBox* with a brief description of his travels in the 1780s.
- Click on the other people in the people map to view their information. You can close each *infoBox* by clicking on its close button. Note that the lines connecting each person indicates whether he/she was linked to Jefferson through diplomatic, social, or commercial connections.
- When done, click on the close button to close the concept map and return to the main Jefferson's Travels view.

WARNING! Do not attempt to get back to the main view by clicking on the Back button of your browser. That will close Jefferson's Travels completely.

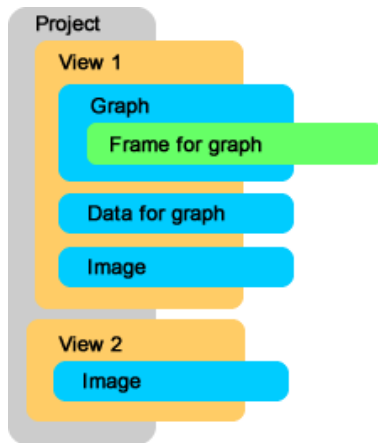
The last feature of the Control Panel is a Search box. This is an advanced feature of VisualEyes, and is described in the VisualEyes Manual.

Elements and Attributes

VisualEyes uses a script format called XML to represent the projects internally. XML is a simple text format for storing information that a computer and a person can understand. If you use the VisEdit tool to create your project, you do not need to know how to format XML, but an understanding of its building blocks is useful:

Elements

Elements are building blocks in VisualEyes that are connected together to create a project. There is one **project** folder, an element that contains your entire project. Many elements, like **project**, can hold other elements. You can think of **project** as the main folder, and the elements it holds as subfolders. Within the **project** folder are five possible elements: *frame*, *tab*, *logo*, *textformat*, and *view*. The *frame*, *tab*, *logo*, and *textformat* elements are self-contained. The *view* element requires additional elements to be nested within it. The **project** element can contain multiple *view* elements, each one displayed as a separate tab in your project, each one containing elements itself, such *resources*, *controls* and *displays* (each box represents an element):



Attributes and values

The attributes control the details of how an element will look or what it will do. If the element was a person, its attributes would be eye color, weight, gender, etc. An attribute is always paired up with a value, for example, *eyeColor* with “blue” and *weight* with “98.”

For example, the **view** element, aside from containing other elements, has an attribute called *title*, which causes the value assigned to it (in this case “My View”) to be written in the



Script text

An element can contain some text that is unstructured. This text is used by VisualEyes to hold GLUE scripts and also for the content in text displays.

Tools: VisEdit

Now that you have explored Jefferson's Travels, you are ready to create your own VisualEyes project. You will do that using VisEdit, an editing tool specifically designed for VisualEyes. With VisEdit, you can create projects without having to work with the underlying software code.

- Point your browser to <http://www.viseyes.org/edit.htm>. The VisEdit screen will load. At the bottom, you will see a gray box saying "Please log in"

- In the Username text box, leave the default user name, guest
- Do not alter or delete the password (hidden by asterisks) in the password box.

Getting started with VisEdit

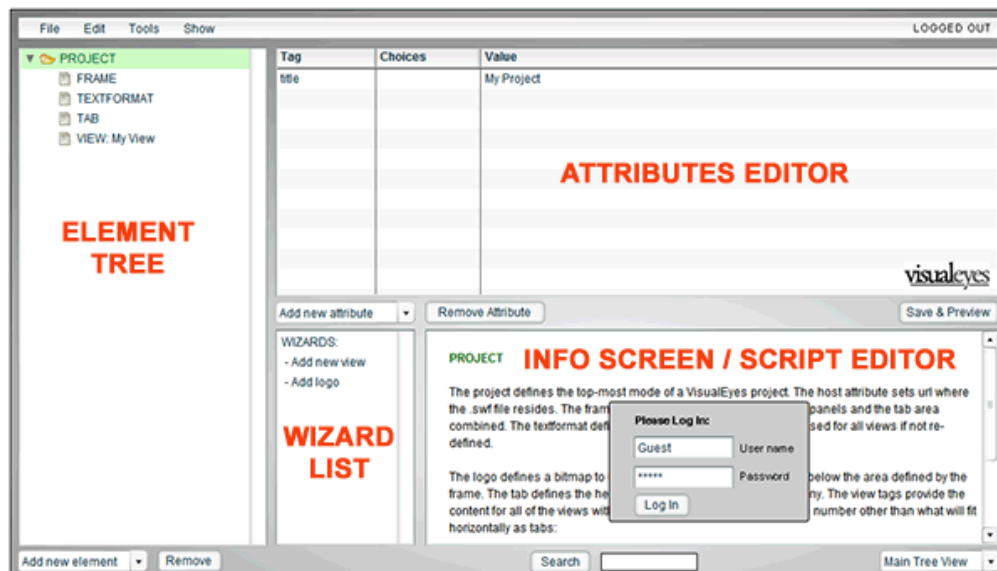
Creating an Account

Begin by creating an account. Click on the link below the UVa logo and fill in the information requested. Then, instead of using "guest" and "blank", use your new username and password. Using your email address is an easy way to get a memorable password. A blank project will load on your screen.

Explore VisEdit

VisEdit is a browser-based interface that allows you to build VisualEyes projects without typing in all of the XML code that VisualEyes uses internally to make the projects. It can be accessed at [VisEdit](#).

- Take a few minutes to examine the screen.



The screen you see when starting VisEdit is the **Main Tree View**. You can see the name of this screen displayed in the lower right corner.

At the top of the screen is a menu bar with four options: File, Edit, Tools, and Show. Your username, samples, appears on the right of the menu bar.

- Click on each to see the menu bar options. You'll learn what they do later.

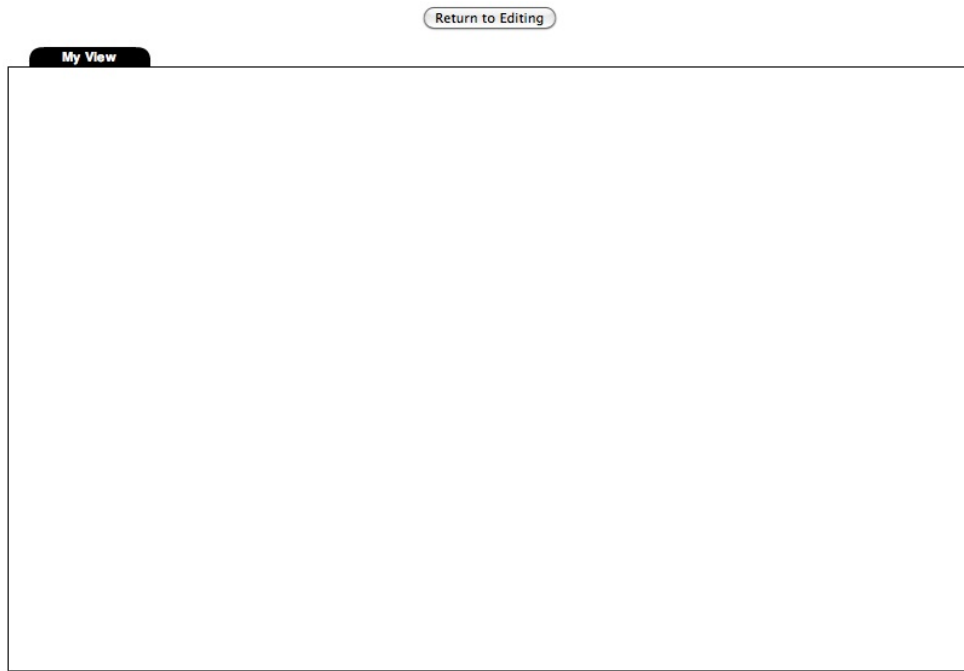
VisEdit's screen is divided into four frames, each dedicated to a specific purpose:

1. The **Element tree** shows all the elements that make up your project and provides tools to add or delete them.
2. The **Attributes editor** displays the attributes and values that go with the currently selected element, allowing you to add, delete, or modify them.
3. The **Info screen/script editor** shows instructional help related to the item you currently have selected and also provide an area to edit GLUE scripts and text displays.
4. Finally, the **Wizards list** provides wizards that walk you through the creation of certain elements, step by step.

Your Blank Project

What do all of these Attributes do? To find out, you can preview what you have created so far.

- Click on the Save & Preview button on the bottom right of the Attributes editor. In a moment, the project will appear.



9/24/10

What you see is your Project as you have defined it, a very, very empty project. The **frame** sub-element sets the size of the box that holds the project. Everything else you add to it will be contained within the **frame**. The **textformat** element defines the size and type of the text. The **tab** element defines the colors for the tab at the top of the frame. And the **view** element defines the title of the view. Right now, you have only one view – with the *title* "My View" – but as you remember from Jefferson's Travels, a project can have multiple views. You access each view by clicking on its tab.

- Click on the Return to Editing button at the top of the screen to return to VisEdit.

WARNING! Do not attempt to get back to the main view by clicking on the Back button of your browser. That will close your project completely, and you will have to sign in again to VisEdit.

Navigating the Element tree

The **Element tree** is where you add, select and remove elements from your project and becomes the main way you work with your project. When you click on an element it will turn green. If the element contains other elements within it, you can see show those elements by

clicking on the arrow to the left - opening the element's folder. Clicking on the arrow again will close the folder. Clicking on an element selects that element as the current element to:

- Show whatever attributes added to that element in the **Attributes editor**.
- Show a description of the element on the **Info screen**.
- Show possible attributes and their default values that on the **Info screen**.
- Show any wizards for adding new elements in the **Wizards list**.
- Add elements that can be added to this element in the **Add new element** button.
- Allow it to be deleted by clicking the **Remove** button.

Back to our sample project. You can see the subfolders by clicking on the arrow to the left of ***project***.

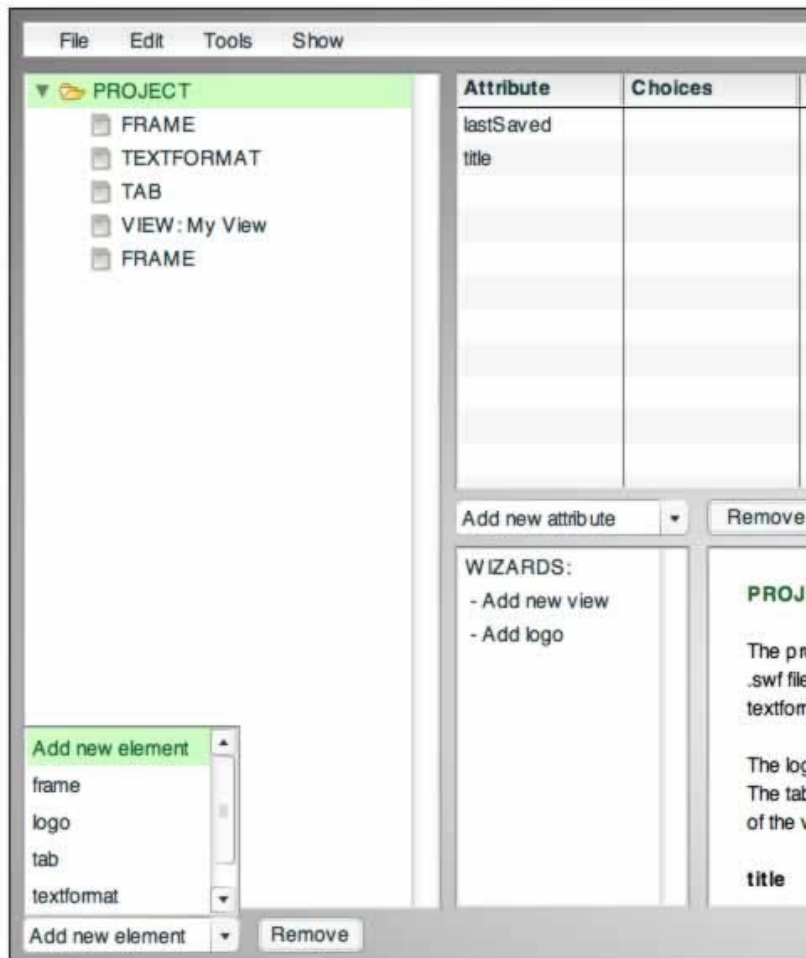
- Click on the arrow. A list of sub-elements appears: ***frame***, ***textformat***, ***tab***, and ***view***.
- Click on each of the sub-elements. As you do, the set of attributes associated with that element appears in the Attributes panel. The material presented in the Information panel changes as well.
- Take a moment to click on the ***view*** element. Note that it has one attribute, *title*, with the value "My View".

The Add new element list box at the bottom of the Element tree will always display the available sub-elements for a highlighted element.

- Click on ***project*** in the Element tree.

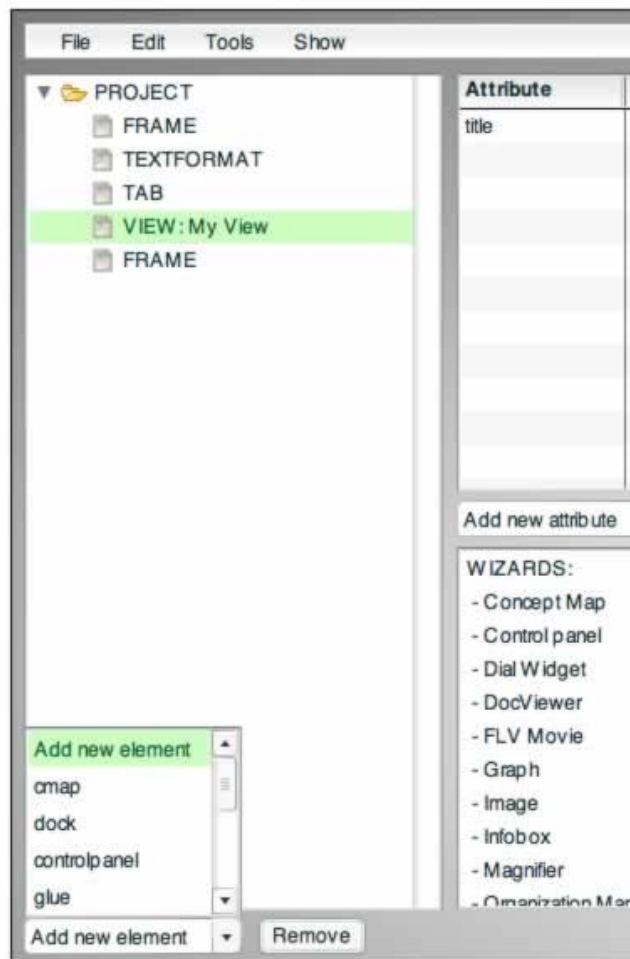
You can find the list of available sub-elements by clicking on the Add new element list box at the bottom of the Element tree.

- Click on the Add new element list box. You will a list of the available sub-elements for ***project***: ***frame***, ***logo***, ***tab***, ***textformat***, and (when you scroll down) ***view***.



Some of the sub-elements have sub-elements of their own.

- Click on the **frame** sub-element, then on the Add new element list box. You will see that it is empty: that is, you can not add any sub-elements to **frame**.
- Do the same for the **textformat** and **tab** sub-elements. They do not have any sub-elements, either.
- Click on the **view** sub-element, then on the Add new element list box. As you can see, there are many available sub-elements for **view**. A view presents a set of features like maps, graphs, and information boxes, so the list of available sub-elements allows you to specify the features to include in that view.



The **remove** button removes the currently selected attribute from the element.

Navigating the Attributes editor

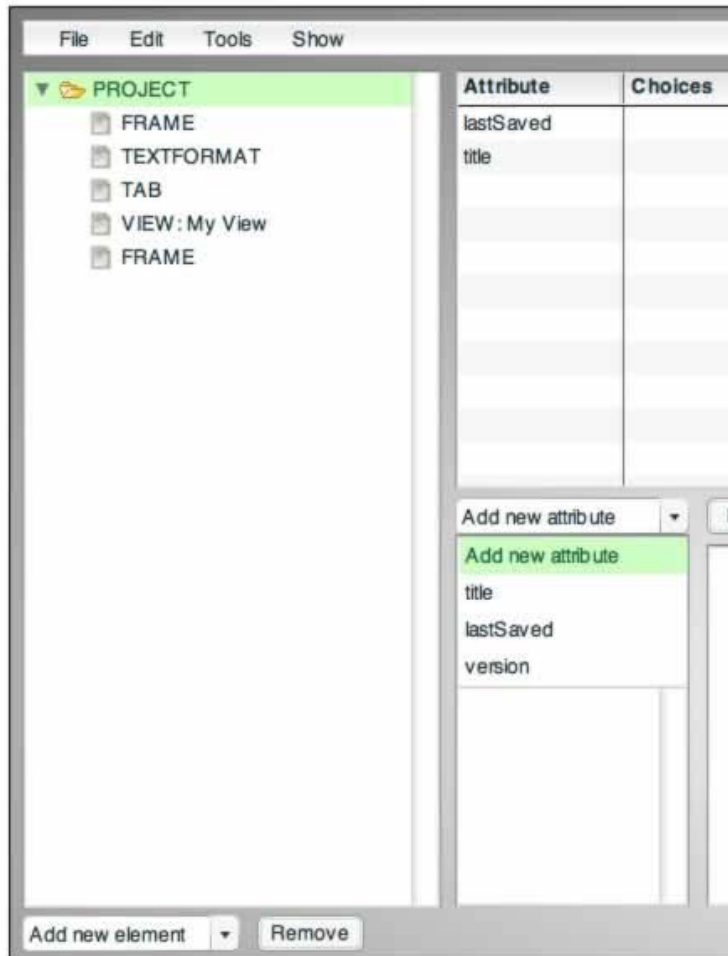
The **Attributes editor** shows the attributes already added to the currently selected element. Each attribute appears on its own line, showing the name, possible values, and the actual value set. Depending on the kind of attribute, the options column will display one of three ways:

1. If setting a text or numeric value, it will be blank and the values column is used to type the value directly.
2. If setting a color, the options column will be filled with that color. Clicking on it brings up a color picker dialog. The color's RGB value appears in the values column and can be edited directly.

3. If there is a list of options, you can select one from the drop-down button. The value can also be edited directly in the value column.

The **Add new attribute** drop-down button adds new attributes to the element.

- Click on the **project** element in the Element tree.
- Click on the Add new attribute list box. You will see a list of the following attributes: *title*, *lastSaved*, *version*.



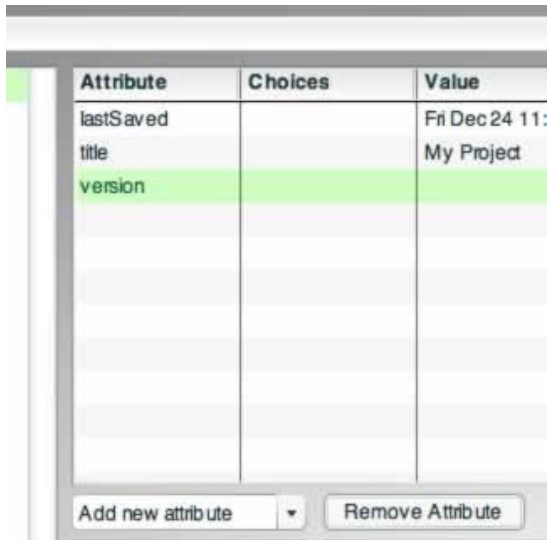
- The first two appear automatically in the Attribute editor. Note that there is now a date and time next to *lastSaved*: it is the date and time you clicked on the Save and Edit button.

You have the option to add the version attribute.

- Click on *version* from the Add new attribute list box. The attribute *version* appears in the Tag column.

For now, you don't need it. To remove an attribute, click on it to highlight it, then click on Remove Attribute.

- Click on *version* if it is not already highlighted, then on Remove Attribute.



Attribute	Choices	Value
lastSaved		Fri Dec 24 11:
title		My Project
version		

At the bottom of the table, there are two buttons: "Add new attribute" (with a dropdown arrow) and "Remove Attribute".

The attribute is removed.

You follow the same type of procedure to add and remove elements from the element panel.

- Practice this by clicking on **tab** from the Add new element list box, and adding it to the element panel
- Click on Remove to remove it.

File, Edit and Tool Menus

VisEdit has a menu bar across the top containing options similar to desktop applications to save and load project, undo/redo, and access to tools:

FILE

- **New** - start a new project. When you start a new project the screen will be very bare bones. Creating a new project creates a new project ID number. This ID is how each project is stored.

- **Load Project** - a list of available projects is visible in the Information Box. Click on the project you want it will load, replacing your current one.
- **Load by ID** - a box will pop up, you enter the id of a project and it will load.
- **Save** - saves the current project to the VisualEyes server.
- **Save As** - saves the current project with a new project ID.
- **Show my Data Files** - Shows a list of your data files
- **Preview** - shows you what your project looks like without saving any changes. Click the **Return to Editing** button at the top of the page to get back to VisEdit. You can also reach this screen by clicking the **Save and Preview** button.

EDIT

- **Undo/Redo** - your ability to undo and redo actions is almost unlimited. Every time you undo an action your project returns to its last version. Redo redoes your most recent undo.
- **Copy/Paste** - you can copy and paste any element. This would be useful if, for example, you wanted to create more than one tab with similar attributes. The element and any sub-elements contained within its folder that are copied and pasted will be added on the same level of the tree as the element you copied. The pasted element will sit just below the element that is currently highlighted.
- **Move Up/Down** - click on an element or attribute you wish to move and click either Move Up or Move Down. The element and any elements contained within it will move up or down in the tree remaining on its same level of the tree. If you are moving around view folders, the order in which they are shown in the element (top to bottom) tree will be the order they are shown in the project preview (left to right).

TOOLS

- **Convert data/AI file to XML** -Converts/uploads files to the server.
- **Upload CSV file to server** -Uploads CSV files to the server.
- **Upload KML file to server** -Uploads KML files to the server.
- **Upload local XML as a project** -Uploads XML project to the server.

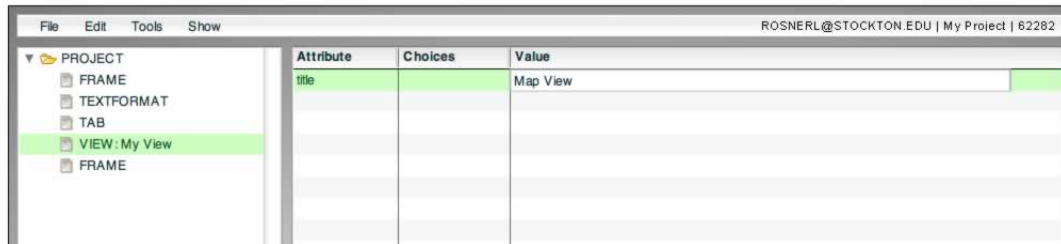
See the Tools section in Appendix for more information about options in the tools menu.

Editing a Project

When you start a new project, it is created with default elements and attributes. You can customize the elements to fit your own ideas. You will practice this by editing the view title, the

words that appear on the project tabs. As you saw, the current *title* for **view** is "My View," but that is not very descriptive. Follow these instructions to replace it with the *title* "Map View".

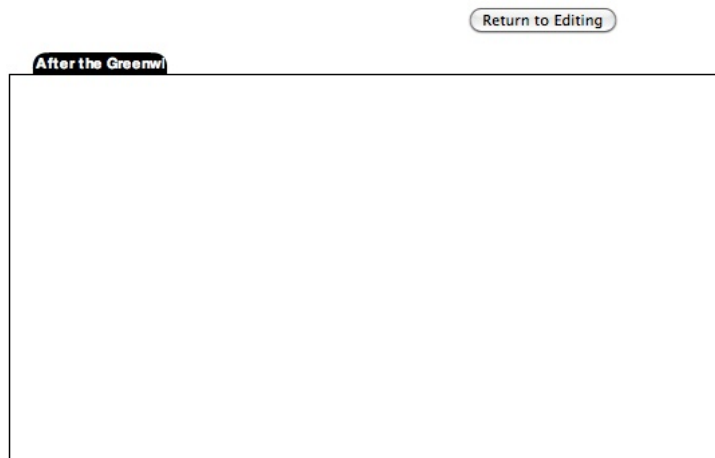
- Click on **view** in the Element tree to highlight it (if the only element you see is **project**, then click on the arrow to the left to display the **project** sub-elements, then click on **view**).
- In the Attribute editor, you should see the tag *title*, with the value "My View".
- Click on "My View" to highlight it. The value becomes editable text. You can change it to "Map View".



- Click anywhere outside the text to remove the highlight.
- Click on the Save & Preview button. The *title* for **view** appears as "Map View".

What if you wanted a more descriptive title? Something like After the Greenwich Tea Party?

- Click on the Return to Editing button.
- Click on **view** in the Element panel, and edit the *title* to "After the Greenwich Tea Party"
- Click on Save & Preview to see the changes.

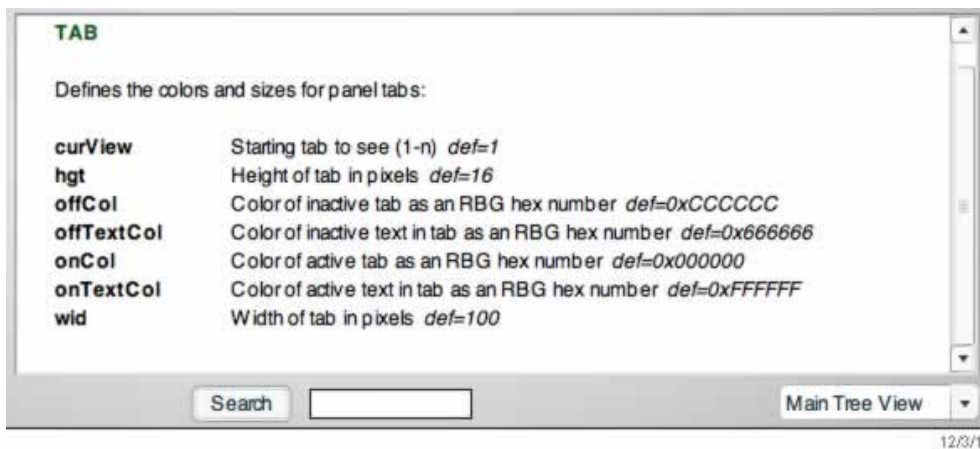


Oh dear. That didn't work out very well. Most of the new title was chopped off. Why?

- Click on the Return to Editing button to investigate the problem.

The reason why the new title doesn't appear is that the tab is too small to hold it. The **tab** element is created with default settings. You can see the settings by looking at the **tab** attributes.

- Click on **tab** in the Element Tree to highlight it, and then look at its attributes in the Attribute Navigator. Right now, none appear.
- But if you look at the Info screen, you can see that it has a number of pre-set attributes.

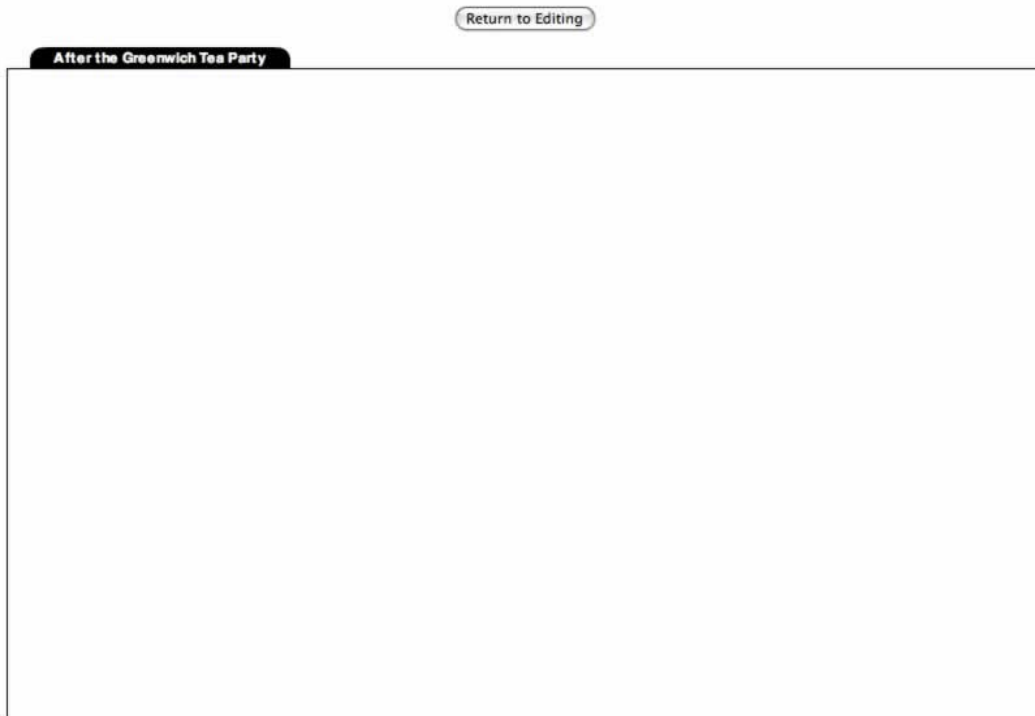


These include *hgt*, for setting the height in pixels, *offCol*, for setting the color of a tab that has not been clicked on, and *onCol*, for setting the color of a tab that has been clicked on. (Tabs change color to indicate the *active view*, that is, the view displayed "on top" of the screen.) And among the other attributes, you will find *wid*, for setting the width of the tab in pixels.

- Look at the default (specified as *def*). It is set at 100, meaning the tab's default setting is 100 pixels wide.

To change the width of the tab, you will specify the attribute *wid*, with a value of "200" pixels.

- Click on the Add new attribute list box, and scroll down until you see *wid*.
- Click on *wid* to add it to the Attribute editor. The default value is "100".
- Click on "100" in the value column to select it, and edit it to "200".
- Click on Save & Preview. Now the title displays properly.



- Click on Return to Editing to get back to the VisEdit screen.

Saving a Project with a New Name

Your project is saved whenever you click on Save & Preview, but the file is overwritten every time you save it. There is no "undo" button, so if you make changes, but don't like them, you will have to painstakingly undo them step by step. Sometimes you may not realize there is a problem until you have made many changes, which can make the "undo" steps very time-consuming.

One solution to this is to save your project a number of times along the way, with names that will help you keep track of each stage. For this manual, you will be saving your work with a new name for each chapter. That way, if you make a mistake, you can always go back to the project as it stood at the end of the previous chapter.

You will end this chapter by saving your project with the name Chapter 1.

- Click on File from the menu bar, then click on Save as... from the menu. A text box appears with the instructions, Type new title to save project as...
- Type Chapter 1, and click on OK.
- Click on **project** from the Element panel if it is not already highlighted. In the Attribute panel, the title has been changed to Chapter 1.

Your previous project, My Project, still exists.

- Click on File, then on Load project, to see a list of your projects displayed in the Information box. You should see both My Project and Chapter 1.

Congratulations! You have learned how to work with VisualEyes and created your first project. You have learned how to create and edit elements and attributes using VisEdit, and how to Save & Preview your project. You have also learned how to save your project with a new name to protect your change.

Chapter Two

In this chapter, you will learn how to add components to your VisualEyes project, including how to:

1. Load a file
2. Create a new view
3. Delete a view
4. Understand external resources
5. Add a map
6. Understand **GLUE**
7. Add components to a view
8. Add zoom control
9. Use VisEdit to modify components
10. Add a timeline

Loading a file

Once you have created and saved a project, it remains available for you to load and resume work on it. Files you create in VisualEyes are not stored on your personal computer, but rather on a *server*, a computer based at the University of Virginia. You access the files through your browser.

NOTE: It is also possible to keep back-up copies of your work on your own computer by copying the XML files.

- Point your browser to the VisEdit screen and log in, as you did for the last chapter.

Your project should be loaded exactly as you left it at the end of the last chapter. All your elements should be available in the Element tree and at the right of the menu bar you should see your project information: Your username, Chapter 1, followed by the five-digit *ID number* that is a unique identifier for that project

As mentioned in the last chapter, it is good practice to save your project frequently in manageable sections, using an easy-to-remember name for each section. You will therefore save the project now, with the name Chapter 2. That way you will retain our Chapter 1 file as a backup, in case you need it.

- Click on File, then Save as..., and save the project with the name Chapter 2. The file is saved and the new file name appears on the right of the menu bar.

What if you wanted to reload your Chapter 1 file?

- Click on File, then Load project. A list of projects appears in the Information panel.
- Click on the ID number for the Chapter 1 project. Chapter 1 is re-loaded.

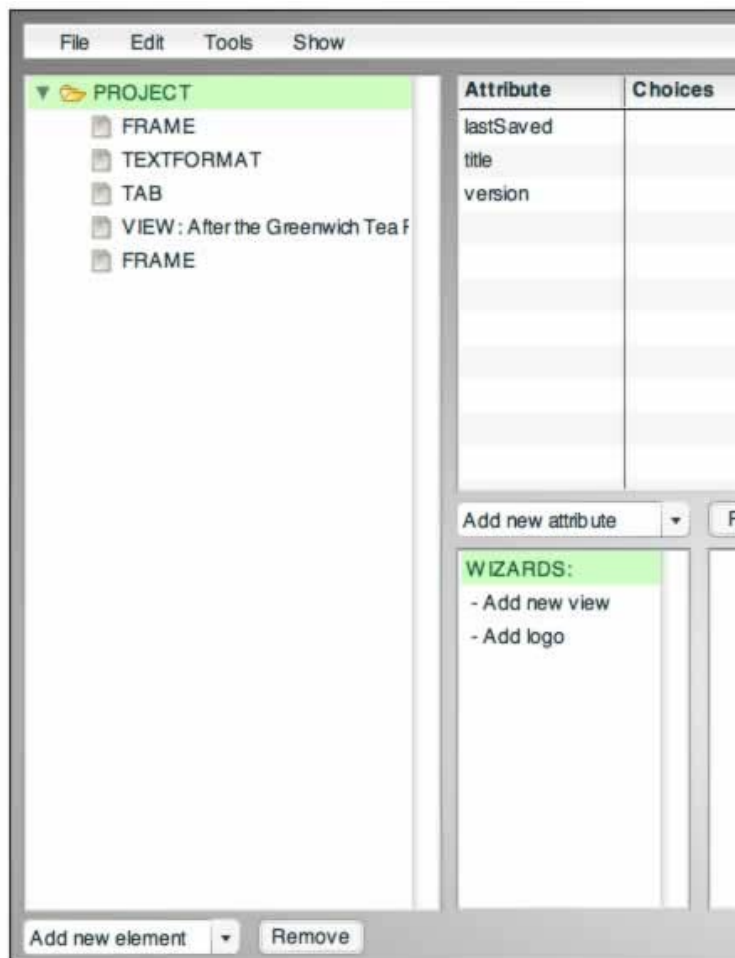
For this chapter, you will be working with the Chapter 2 file, retaining the Chapter 1 file as a backup.

- Click on File, then Load project, and click on the ID number for the Chapter 2 file to load it.

Using Wizards

Wizards are a feature built into VisEdit that walk you through the creation of certain elements and attributes step by step. They are context-sensitive, and are available for frequently used features. For example, in the main project folder you are given two options for using a wizard, Add new view and Add logo.

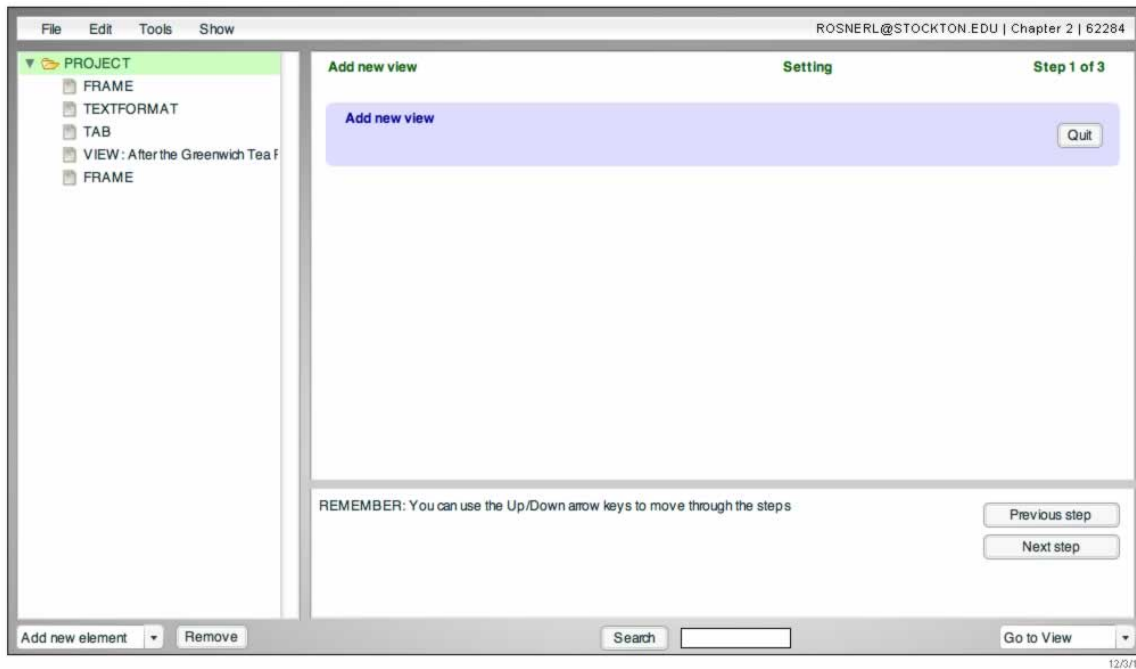
- Click on **project** in the Element tree if it is not already highlighted. The available wizards will be displayed in the Wizard list.



Adding a View

You will use the wizard to add a new view.

- Click on Add new view in the Wizard list. The screen changes to the wizard format, and Step 1 of 3 is displayed.



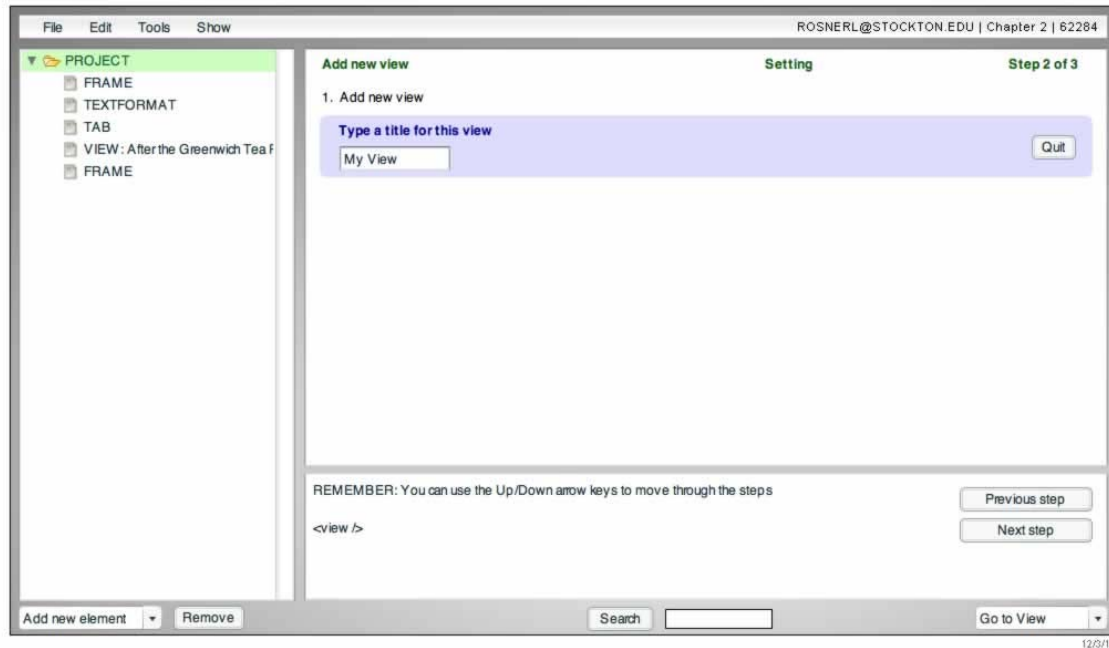
In the top section of the screen is the name of the wizard, Add new view, and an option to press the **Quit** button to return to the main project screen (Main Tree View). You can quit at any time before the wizard sequence is complete to cancel the action.

In the bottom right hand corner are two buttons to help you navigate through the wizard. They are the Previous step and Next step buttons. Clicking the **Next step** button, or hitting the **Down-Arrow** key will start the wizard actions. Each step requires you to make a decision, add a number, add a name, or add a link from a URL or to a data source. Hitting the **Previous step** button or **Up-Arrow** key lets you go back and make changes.

When the wizard has completed its sequence an option to click the **Add** button will pop-up. Clicking it will bring you back to the Main Tree View. You can now view the changes you made by clicking the Save & Preview option. If you want to make adjustments or remove the logo you just added those changes can be made in the Elements tree and Attributes editor.

- Click on Next step. Step 2, Type a title for this **view**, is displayed.

You may notice that a piece of XML code, `</view>`, is displayed in the Info screen. The code is part of a *script*. It is a piece of the XML instructions that VisualEyes software uses to create and name the new view. As you will see in later chapters, VisualEyes uses many different kinds of scripts. For now, just note that when you work with a wizard, the bottom panel is set up to display the code that the wizard creates.



- Enter the title Second View, and click on Next step. Step 3 of 3 is displayed.

This is the final step of the wizard, letting you know the wizard is Done and prompting you to add the view to the project. You may note that another piece has been added to the script in the Info screen, `<view title="Second View" />`.

- Click on Add. The **view** is added to the project.
- Click on Save & Preview to see the changes you've made.

As you can see, your project now has two tabs, corresponding to your two views.

The first one you defined, "After the Greenwich Tea Party, should be *active*, that is, be displayed on top. To make "Second View" active, you can click on the tab.

- Click on "Second View". The text and background color of the tab changes, indicating that it is active.



- Click on Return to Editing to return to the main VisEdit screen.

Follow the same steps to create a third **view**, with the title Third View.

- Make sure **project** in the Elements tree is highlighted, then click on Add new view in the Wizard list. The wizard format is displayed, on Step 1 of 3.
- Click on Next step to get to Step 2 of 3.
- Enter Third View as the title, and click on Next step.
- Click Add to complete the wizard and return to the Main Tree View. The new **view** appears in the Elements tree.

Deleting a View

The easiest way to delete a view is from the Main Tree View. **Be careful, though! Deleting a view will delete all the resources, data, and displays associated with it!**

Since you have not created any features for “Third View”, you can safely delete it.

- Click on VIEW:Third View in the Element Tree to select it.
- Click on Remove at the bottom of the Element tree. The **view** is removed.

Understanding External Resources

Now that you have set up your views, it is time to add features to begin to make them work. First, you will use the Wizards feature to add a map of southern New Jersey from the early 19th century to your project. This map is available online at <http://www.viseyes.org/sampler/NJsouth.jpg>.

In order to add a map to a view, you will find it helpful to understand how VisualEyes works with *external resources*, that is, resources held in files created outside the software. These can be images (photos, documents, maps and more in .jpeg, .gif, or .png from any valid URL), movie clips (FLV Flash video clips on the web, and clips from YouTube), numbers or spreadsheets.

To work with these files, you must first make sure that they are available online, since VisualEyes can't read them on your computer. Generally the URL address is entered as part of a wizard. It can also be added directly with the Attribute editor, as the *src* attribute.

Adding a Map

For this project, you will add the historical map of southern New Jersey to your first view, After the Greenwich Tea Party. (Remember, wizards can be canceled at any time by clicking the Quit button.)

- Click on the “After the Greenwich Tea Party” view.
- Click on “Image” from within the Wizard list. The wizard format will appear, displaying Step 1 of 7, Add new image resource.
- Click on Next step.

Step 2 prompts you for an ID for this resource. An *ID* is the name VisualEyes associates with this resource as you work with it. You'll find it helpful to use short, descriptive names for each resource or feature. The names will not appear in the final project as it is displayed online.

- The default ID is myImage. Change it to NJsouth, and click on Next step.
- Enter the following URL: <http://www.viseyes.org/sampler/NJsouth.jpg>, and click on Next step. Note that each step adds XML code, displayed in the Info screen.

Image

Setting

Step 4 of 7

1. Add new image resource
2. Type an ID for this resource
3. Type URL for this resource

type=image

NJsouth

http://www.viseyes.org/sampler/NJsouth.jpg

Adding GLUE to show resource

Quit

REMEMBER: You can use the Up/Down arrow keys to move through the steps

Previous step

Next step

<resource src="http://www.viseyes.org/sampler/NJsouth.jpg" id="NJsouth" type="image" />

Search

Go to View

- The next step, Step 4 of 7, is titled Adding GLUE to show resource. We'll explain what that means in the next section. For now, just click Next step.
- Step 5 of 7 has the prompt, Should it show up initially? The default is "true". Again, we will explain this in the next section. For now, click Next step.
- Step 6 of 7 prompts you to Type an ID for this GLUE. The default is showImage.
- Change it to showNJsouth and click Next step.
- Click Add to complete the wizard and return to Main Tree View.

Take a few minutes to see what the changes look like in the Main Tree view.

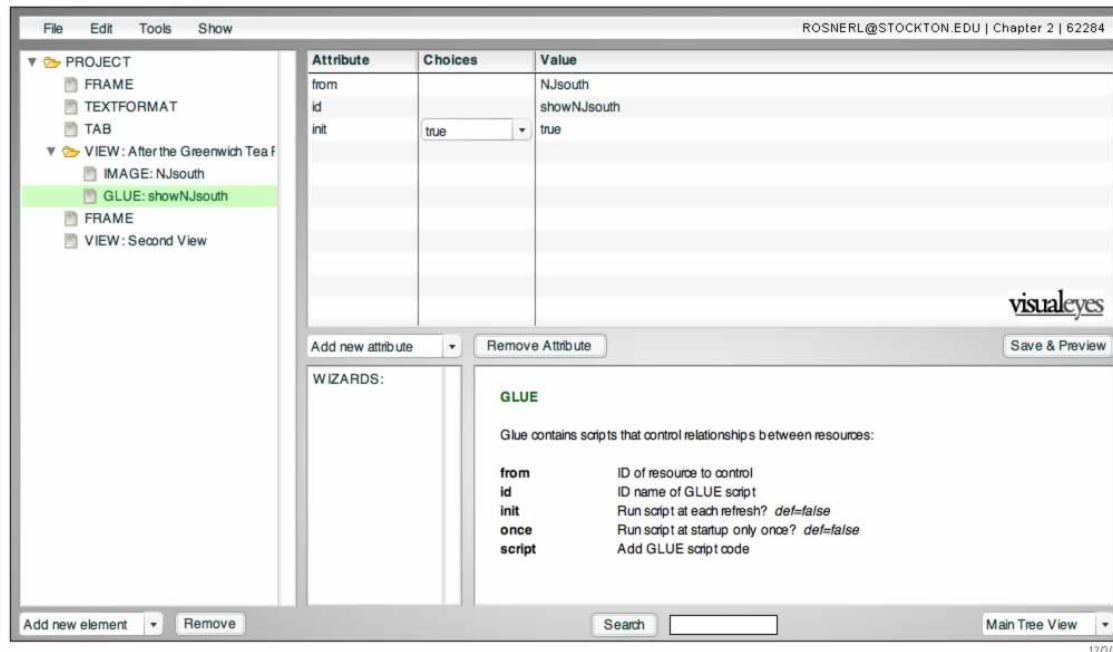
- Note that there is now a folder icon next to your first **view**, and a small right-pointing arrow. This indicates that the view has sub-elements.
- Click on the arrow to see the sub-elements, **image** and **GLUE**.
- Click on IMAGE: NJsouth to select it.
- Take a few minutes to look at the Attributes editor.

Note that **image** has the attributes *src*, *id*, and *type*. The value for *src* is the URL you specified, "<http://www.viseyes.org/sampler/NJsouth.jpg>", and the value for *id* is "NJsouth". The value for *type* is the default, "image". Images can have other attributes as well: you can see them displayed in the Info screen.

The other new element is **GLUE**.

- Click on GLUE: showNJsouth to select it.

It, too, has the attributes you specified: *from* is "NJsouth", *id* is "showNJsouth", and *init* is "true".



But what do they mean?

Understanding GLUE

VisualEyes uses a specific kind of script to identify external resources and add them to projects. This kind of script is called **GLUE**, short for General Language to Unite Events. **GLUE** appears in VisEdit as an **element** in the Element tree and it is the heart of making interactive visualizations. **GLUE** elements can do a wide range of things. Their most common functions are:

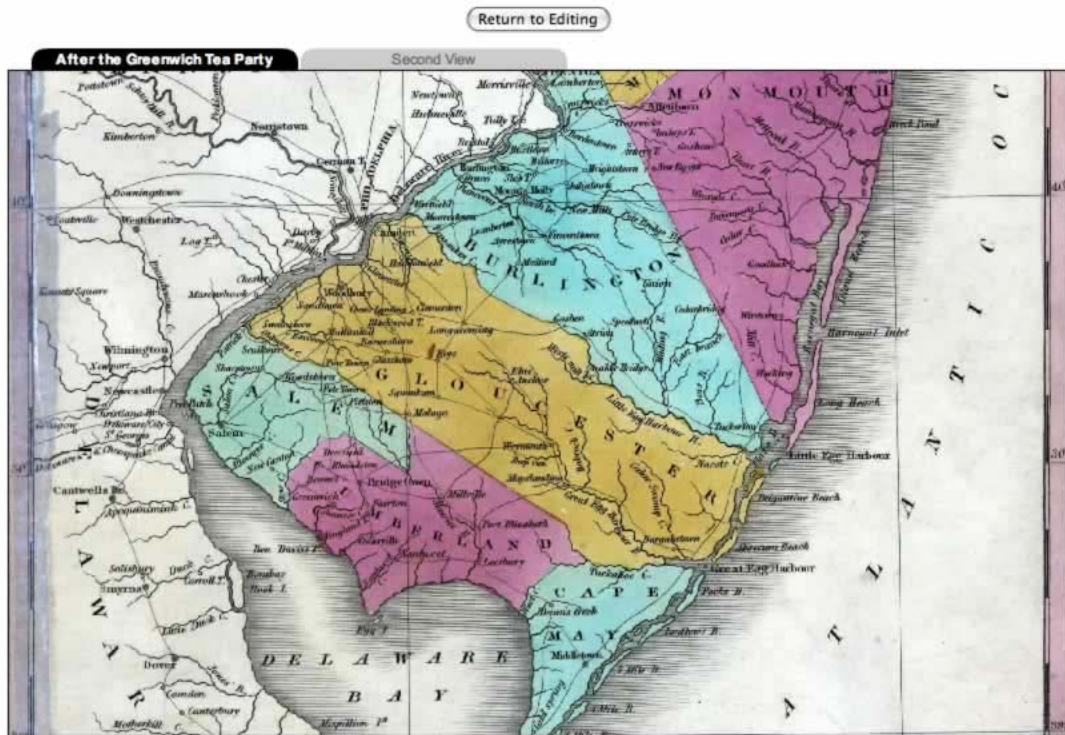
1. To cause resources, such as images, paths, and charts, to show up on the screen, automatically or on command.
2. To connect the data resources to viewers, through features such as display tables, popup windows, charts, and data-driven maps.

For now, you don't need the technical details of how **GLUE** works with maps. All you need to know is the basic idea behind it: You first identify the map you want to display by giving it a unique ID, and then you instruct VisualEyes to **GLUE** it to the **view** in which you want it to appear. Note that you do not glue a map to an entire **project**, but only to a specific **view**. That means that if you want to use it in more than one view, you must identify it and *glue* it to each one separately.

The image wizard you used took care of these details for you. It automatically linked the map to the correct **view**, After the Greenwich Tea Party. It prompted you for an *id* for the map, so the resource can be written into the **GLUE** script correctly, and connected it to the correct URL (*src*) where the image can be found. It prompted you to create an *id* for the **GLUE** itself, so you can use it in this and other scripts. It prompted you for the *id* of the image you want to **GLUE** to the **view** (the *from* attribute), and asked you whether the image should show up as soon as the project is displayed (*init*="true"). Were *init* to be set to "false" this **GLUE** would need to be triggered by a particular event, like clicking a button or reaching a certain date on a timeline. In this case, you want the map to be displayed when the program is loaded, so you can leave the default setting, which is "true." This means "Yes, I do want the map to show up when the program is first started."

To see the results,

- Click on Save & Preview. The map appears on the "After the Greenwich Tea Party" tab.



- Notice that you can click on the map and drag it around. This is because the map is slightly larger than the frame. (The map can be pegged in place by adding a *pan* attribute to the **view** element you are working in and changing the default setting to "false".)
- Click Return to Editing to return to the Main Tree View.

Adding a Zoom Control

Now that you have a map of southern New Jersey, wouldn't it be useful to be able to zoom into it? You can add a zoom control by using a View wizard.

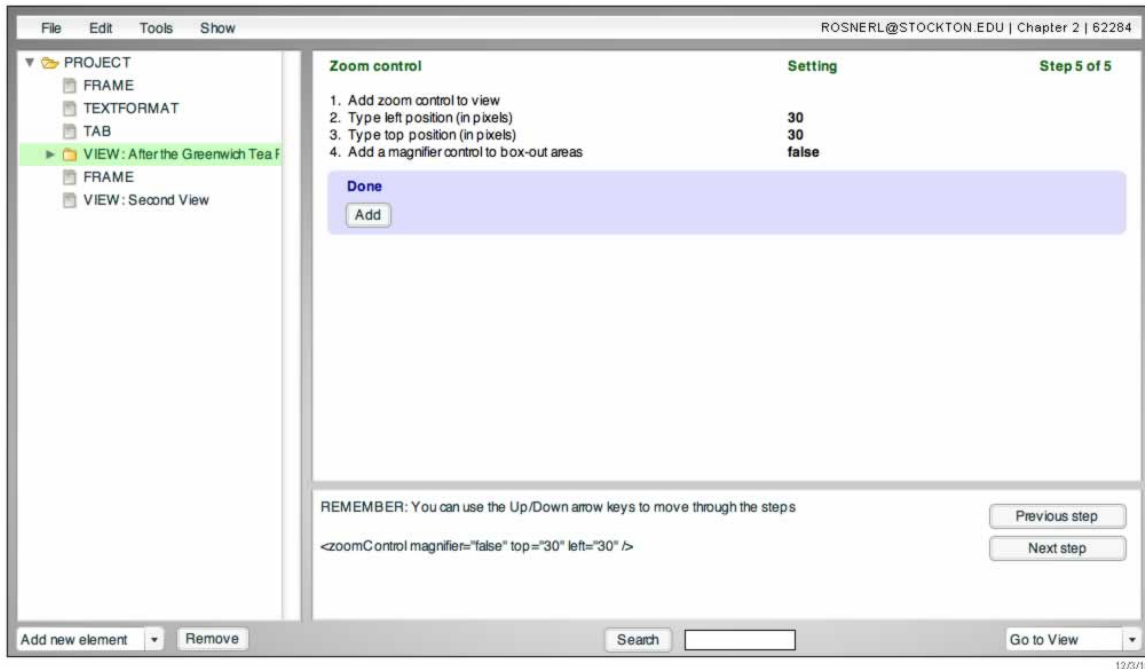
- In the Elements tree, click on the VIEW: After the Greenwich Tea Party folder to select it.
- In the Wizard list, click Zoom control to select it. The wizard will display Step 1 of 5.

Step 1 for all wizards just asks whether you wish to start the specified wizard. In this case, it asks whether you want to add a zoom control.

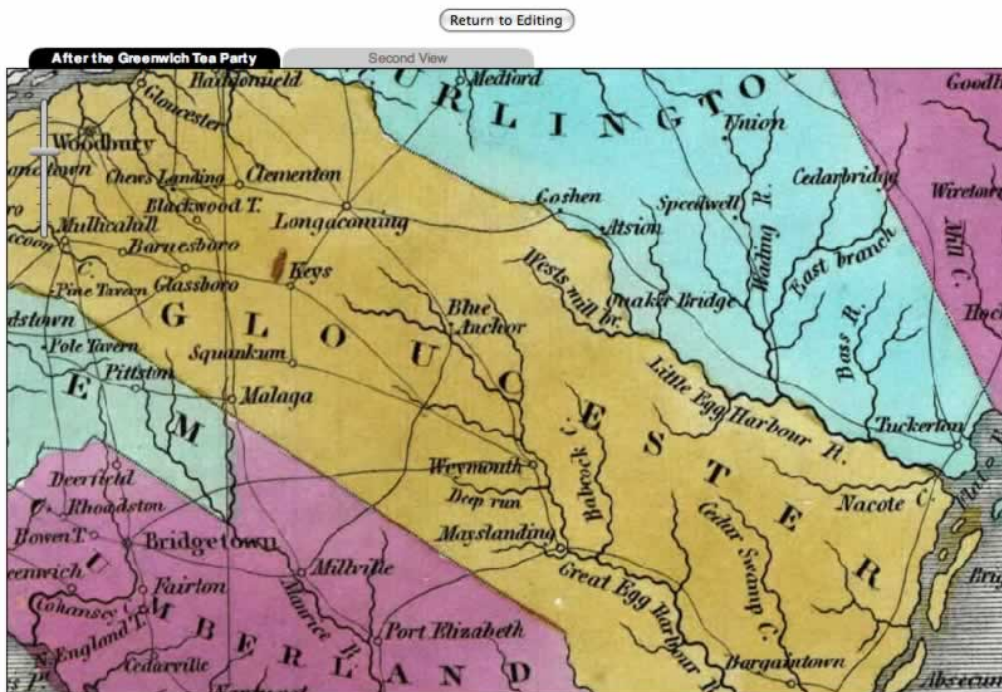
- Since you do, click Next step.

For this wizard, you will keep all the default values:

- For Step 2, Type left position (the position from the left side of the screen), in pixels, the default is "30". Click Next step to accept it.
- For Step 3, Type top position (the position from the top of the screen), in pixels, the default is "20". Click Next step to accept it.
- For Step 4, Add a magnifier control to box-out areas, the default is "false". Click Next step to accept it.
- For Step 5, click Add to complete the wizard. Note the script for the zoom control in the Info screen.



- Click on Save & Preview to view your changes. The zoom control appears at the upper left corner of the map. You also have the ability to click and drag the image.



- Click on Return to Editing.

Adding a Timeline

The last feature you will add in Chapter 2 is a timeline. It is one of the most important controls in VisualEyes, allowing you to create dynamic presentation of material linked to specific dates. Again, you will be using a **view** wizard.

- Click on the VIEW: After the Greenwich Tea Party folder to select it.
- In the Wizard list, click Timeline to start the wizard.

Note that you are at Step 1 of 12. This is a long wizard, so you will find it helpful to remember that you can use the Previous step button to retrace your steps, or the Quit button if you want to quit and start over.

- Click Next step to move to Step 2.
- In Step 2, Type starting date, enter the start date as "12/22/1774".

This is the format for December 22, 1774, the date of the Greenwich Tea Party, in which patriots in Greenwich, NJ seized a shipment of British tea from the house of a local loyalist and publicly burned it. It is considered the first conflict between patriots and British authorities in New Jersey.

- Click on Next step.
- In Step 3, Type ending date, enter the end date as "04/02/1783".

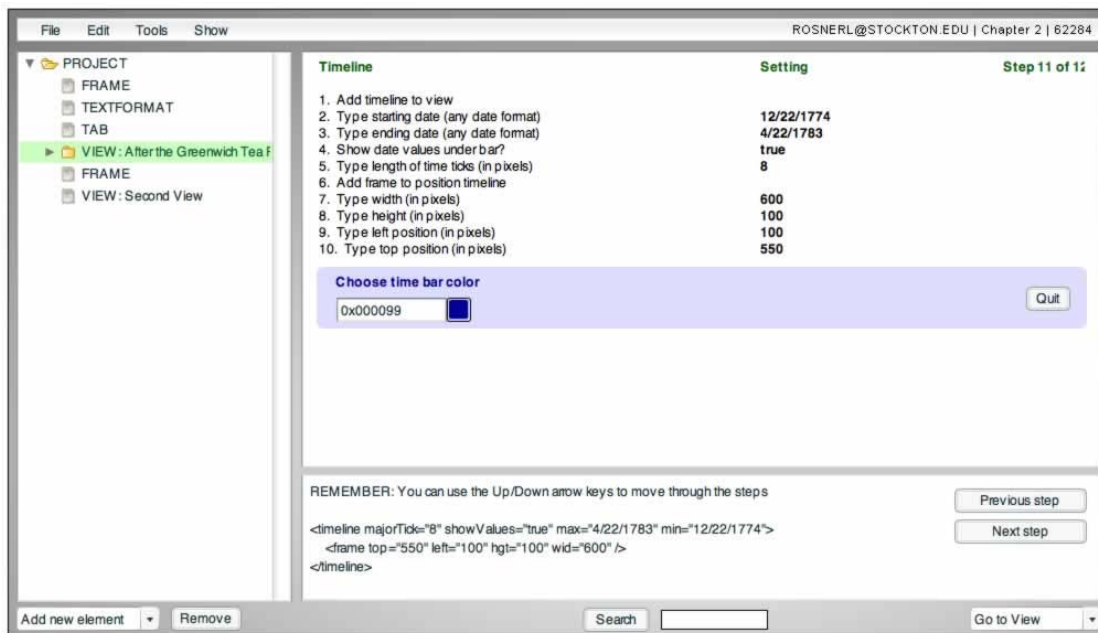
This is the format for April 2, 1783, when Captain John Stewart shot the notorious Loyalist Captain John Bacon near Tuckerton, considered the last skirmish of the war in New Jersey.

- Click on Next step.
- Accept the default setting for Steps 4 through 10.

These control the physical features of the timeline, including the tick length, frame, width, height, and position.

In Step 11, Choose time bar color, the default is gray, but you are going to change it to blue.

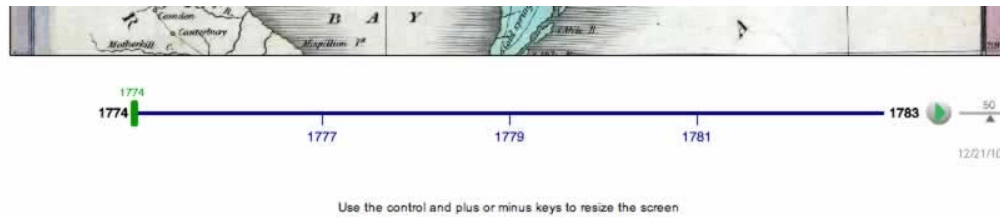
- In Step 11, click on the gray box. A color palette will appear.
- Click on the box with your favorite shade of blue. The bar color is now blue. Note the script in the Info screen.



- Click Next Step to get to Step 12..

- Click Add to complete the wizard.
- Click on Save & Preview to view the project.

Well, the timeline is there, but the wizard did not do exactly what you'd expected. Only the years are showing, not the days or months!

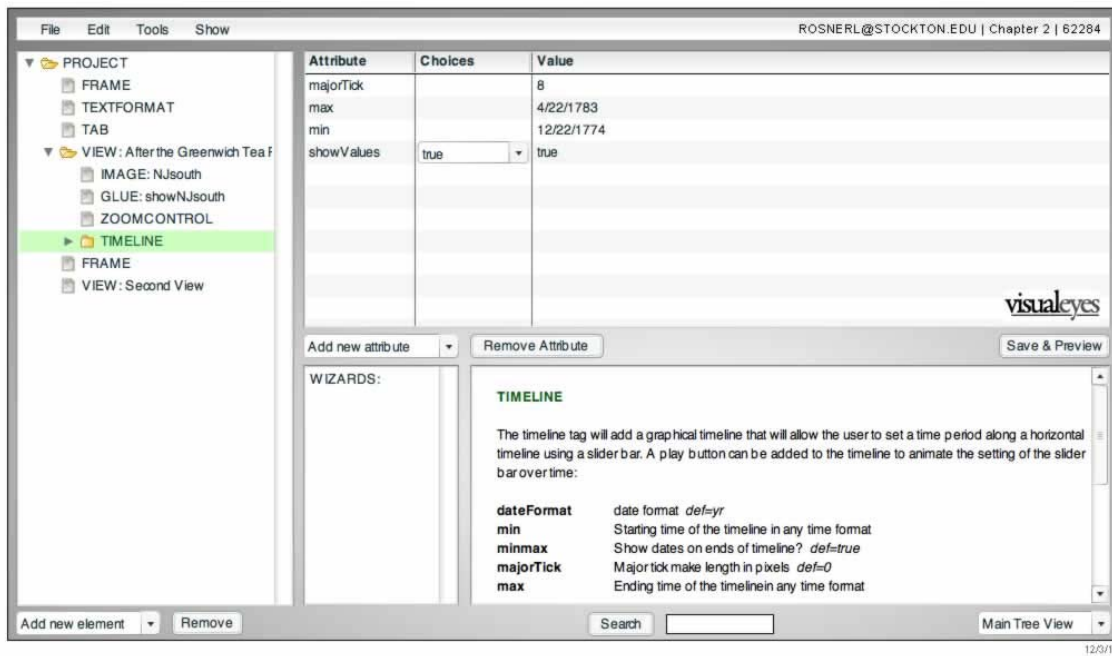


- Click on Return to Editing.

Modifying a feature

To make the dates display properly, you must add an attribute called *dateFormat* to the timeline.

- In the Element tree, click on the arrow next to VIEW: After the Greenwich Tea Party, to display the sub-elements.
- Click on the sub-element **timeline**. You can see all the attributes you created with the wizard: *majorTick*, *showValues*, *max*, and *min* in the Attribute panel.

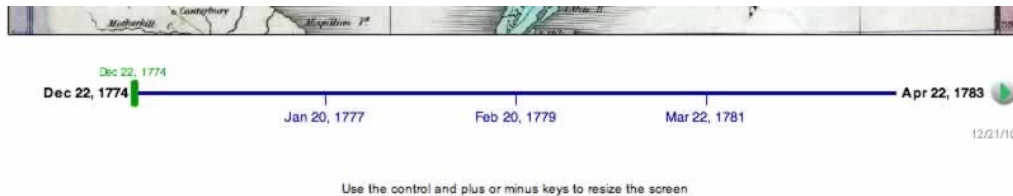


- In the Attribute editor, click on the Add new attribute list box, and then click on *dateFormat*. The attribute is added.

The default value is listed as "yr", which explains why the timeline only displayed the years.

- Click on the menu under Choices to see the other possibilities.
- Click on the last choice, "mo,dy,yr". The value changes to "mo,dy,yr".
- Click on Save & Preview.

That's better! Now the dates are properly displayed in month, day, and year format.



- Click on Return to Editing. You have completed Chapter Two.

Congratulations! You have successfully used wizards to add an **image** (map), **zoom control**, and **timeline** to your **view**. You have also learned how to navigate between wizards and the Main Tree View. Finally, you have been introduced to **GLUE** and scripts. You will learn more about all these features in the next chapter.

Chapter Three

In this chapter, you will learn how to create paths to display events that change over time and space:

- How to:
 - Understand **paths** and **dots**
 - Understand screen redraw
 - Create a simple **path** with the **Path** wizard
 - Positioning **dots** using x and y coordinates
 - Edit a **path** in VisEdit
 - Understand the time attribute
 - Understand the date attribute
 - Use Copy and Paste in VisEdit

Understanding **Paths** and **Dots**

In Chapter Two, you created a timeline from December 22, 1774 to April 22, 1783. If you click on the Play button, you can watch the timeline slider move through those dates. In this chapter, you will learn how to harness the power of VisualEyes to connect that timeline to events on the map. You do this by creating a **path**, constructed from **dots** that connect a specific date on the timeline with a specific point on the map. The **path** allows you to display routes or events through time.

In order to understand how **paths** work, you will start by taking a closer look at the Jefferson's Travels visualization.

- Point your browser to <http://www.viseyes.org/show/?base=jt>. After a few moments, the Jefferson's Travels visualization will load.
- Use the timeline slider to position the timeline on April 8, 1786.



As you can see, VisualEyes has drawn an orange line on the map, connected by icons in the shape of buildings. This is a very easy and intuitive way to display the route Jefferson traveled during this period, with the icons representing his stops along the way. The VisualEyes element that defines the route is called the *path*. It is drawn by defining a set of **dots**, indicated by the icons.

- Move the slider to April 28, 1786.

Again, it is very easy to understand the display. The orange **path** represents Jefferson's travels from the time he arrived in England until he left two months later. The **dots** show his stops along the way. You can note that the the infoBox in the lower right corner displays Jefferson's memorandum entries, which are also linked to specific dates.

What's going on behind the scenes to make this work?

Understanding Screen Redraw

VisualEyes projects are highly interactive. For that reason, the screen constantly needs to be redrawn to reflect changes implemented by the user. Each time you click on the screen and drag the map, scroll the timeline, or move the zoom control, you send a command to VisualEyes

to adjust itself to your response. This is called a *screen redraw*, and it is one of the most valuable features of VisualEyes.

Some of the instructions for the screen redraw are built into the program. Others you can add yourself. For example, when you added the map to your view, you specified that one of the attributes for the **GLUE** element, *init*, was set to a value of "true". This **GLUE** script tells VisualEyes to load the map as soon as the project loads, rather than waiting for a command from the user.

When you create **paths** and **dots** to represent change over time, you create a different kind of **GLUE** element, one that links a specific kind of display – a line and/or a **dot** – with a specific date on the timeline. Pressing the Play button sets the timeline in motion. Behind the scenes, the **GLUE** goes to work. It's as though the **GLUE** was following a set of instructions: "Attention VisualEyes! When the timeline reaches the specified point, draw the **path** and/or **dot** on the screen." The instructions continue – for as many paths and/or **dots** as you've specified – until the timeline reaches the end.

We'll discuss some of the technical aspects of timeline control as we go along. For now, you will use a combination of wizards and VisEdit elements to create a simple **path** showing some of the movements of American troops in New Jersey during the Revolutionary War.

Start out by loading VisEdit for your project.

- Point your browser to the VisEdit screen and log in.
- Click on File, then Save as..., and save the project with the name Chapter 3.

Creating a Simple **Path** with the **Path Wizard**

There are several different ways to provide VisualEyes with the information needed to draw a path. You will start out with the simplest, the **Path** wizard.

- In the Element tree, click on the View: After the Greenwich Tea Party folder to select it.
- In the Wizard List, click Path to start the wizard.
- In Step 2 you are prompted for a unique ID to use for this path; enter "pathNJsouth".
- Accept all the default values for steps 3 through 10. These will specify the color of the path, two dots the size and width of the path and dots, and the **GLUE**.
- In Step 11, you are prompted for a unique ID to use for the **GLUE** to display the path; enter "showpathNJsouth"



- In Step 12, click “Add” to complete the wizard and return to the Main Tree View.
- Click on Save & Preview. You can see the first **dot** in the upper left corner of the map. It is a black square to the right of the zoom control.
- Press the play button. As the timeline moves, the **path** will extend from the first **dot** to the second.

This **path** may work perfectly, but it is not very useful. The **path** starts somewhere in Pennsylvania and ends before it ever reaches New Jersey. How can you edit the **path** so it follows the actual route of Revolutionary War battles?

Positioning Dots using X, Y Coordinates

In order to display actual places on the map, you must specify the position of the **dots** on the map. You do this by specifying the x and y coordinates for the map location. The map, like other computer images, is made up of pixels, and each pixel has a unique x (horizontal) and y (vertical) location.

When you created your **dots** with the **path** wizard, the first was given the default x, y setting of (100,100), corresponding to the point 100 pixels counting from the left edge (the x coordinate) and 100 pixels down from the top edge (the y coordinate). The second **dot** had the default x, y setting of 400,400, corresponding to the point 400 pixels from the left edge and 400 down from the top edge.

You can see the x,y coordinates for any point on the map by holding down the Alt key (or the shift key on a Mac) while clicking on the point. The coordinates will appear in the lower right of the project, just under the play button of the timeline.

Use this technique to view the coordinates for **dots** you've created.

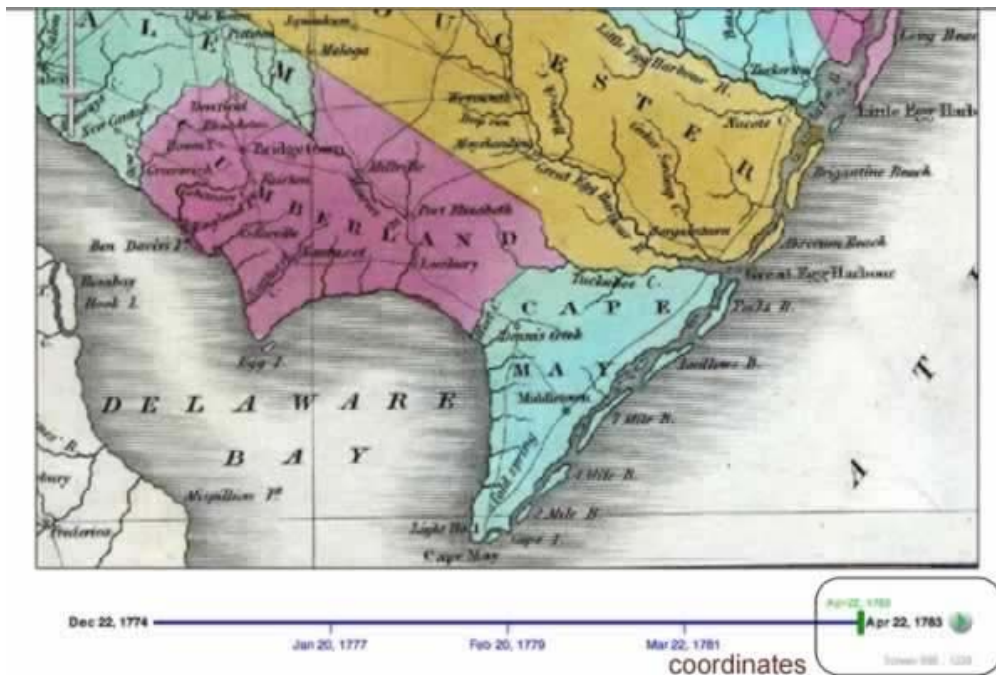
- You should still be previewing the project. While holding down the Alt (or for the Mac, the Option key) button, click on the first **dot**. In the lower right corner of your screen (underneath the Play button), you will see something close to Screen 98,97. The bar is 20 pixels wide and is centered on the specified point, so the coordinate values will be somewhere between 90 and 110.

Now, do the same for the second **dot**.

- Hold down Alt (or Option) while clicking on the second **dot**. In the lower right, you will see something close to Screen 398, 402. These values will be within a range because the bar is 20 pixels wide. Again, the actual values will range between 390 and 410, because the bar is 20 pixels wide.

To edit the **dot** information to match the actual movements of the troops during the war, you will modify their coordinates to match those for Greenwich – site of the Greenwich Tea Party, considered the first action by New Jersey patriots against the British – and for Cape May, site of the last casualty on New Jersey soil.

- Find Greenwich on the map. It is on the Delaware River in Cumberland County. You may have to drag the map and zoom in to find it. Hold down Alt and click on the city. The coordinates displayed should be close to 585, 790.
- Now, find Cape May on the map, on the southern tip of New Jersey. Again, you may have to drag the map and zoom in to find it. Hold down Alt and click on it. The coordinates displayed should be close to 890, 1220.



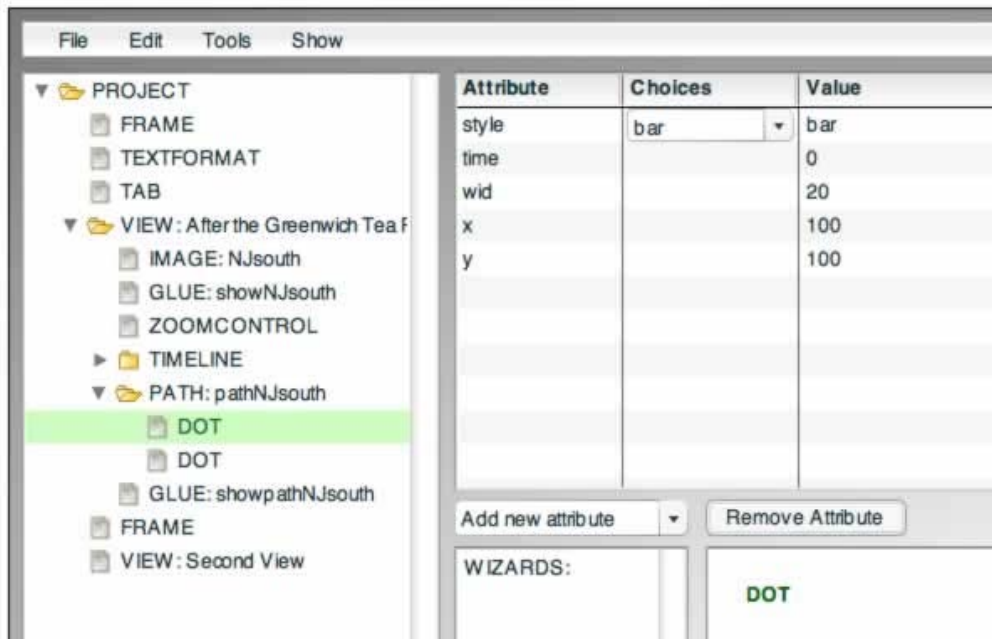
Note these coordinates. The first is the x coordinate, and the second is the y coordinate. You will need them to modify the path.

- Click on Return to Editing.

Editing a *Path* in VisEdit

As you saw in the last chapter, you can use VisEdit to modify a component you have created with a wizard.

- Locate the PATH:pathNJsouth element. It will appear under VIEW:After the Greenwich Tea Party in the Element tree. You may have to click on the arrow to the left of the **VIEW** to display it.
- Click on the arrow next to PATH:pathNJsouth to display the two **dots** you created.
- Click on the first **dot** in the Element tree. Its attributes are displayed in the Attribute editor.



Note that the values for *x* and *y* are each "100". You want to change that to the coordinates for Greenwich:

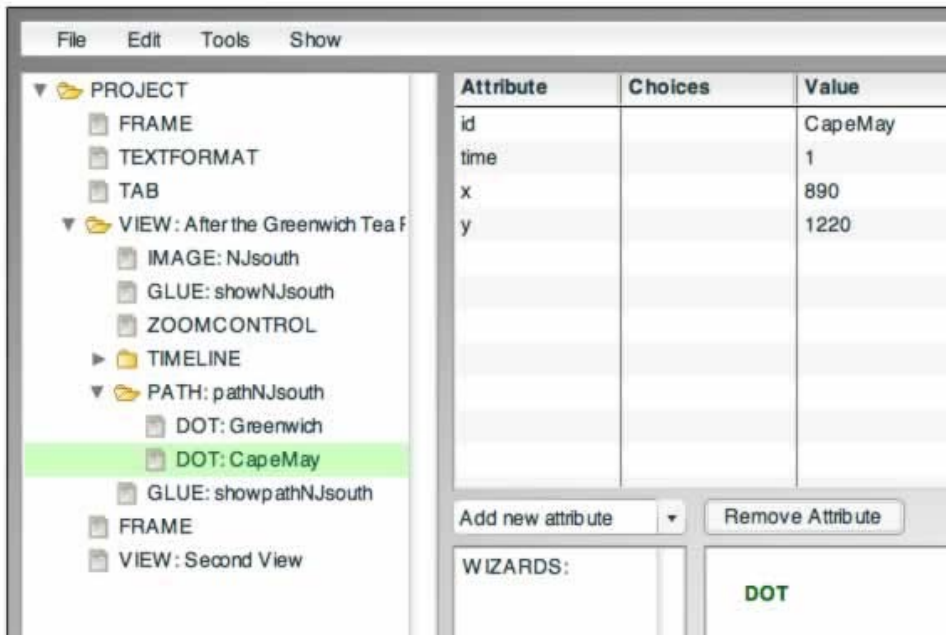
- Edit the *x* and *y* values: *x*="585" and *y*="790".

Since you will be creating a number of **dots**, you'll find it helpful to identify them. Give this **dot** the *id* "Greenwich".

- In the Attribute editor, click on Add new attribute, scroll down until you see *id*, then click on it.
- Enter the *id* value "Greenwich". Note that the *id* must be **one-word**.
- Click on the second **dot**. Its attributes appear in the Attribute panel. Note that the *id* "Greenwich" now appears next to the first **dot** in the element panel.

Now, edit the coordinates, and create an *id*, for the second **dot**.

- Edit the *x* and *y* values: *x*= "890" and *y*= "1220".
- Create an *id* attribute, and enter the *id* "CapeMay". Note that the *id* must be **one-word**.
- Click on any element. The *id* "CapeMay" appears next to the second **dot** in the Element panel.



- Click on Save & Preview.
- Click on the Play button.

The location of the **dots** is now correct, but their timing is not. The Greenwich Tea Party took place on December 22, 1774, but the skirmish at Cape May took place on June 29, 1776, not in 1783. You can correct this with a little additional editing.

- Click on Return to Editing to return to the Main Tree View.

Understanding the Time Attribute

- Click on the Greenwich **dot** in the Element tree to display its attributes. Note that it has an attribute, *time*, that has a value of "0" (zero).
- Click on the Cape May **dot**. Note that it, too, has the attribute *time*, with a value of "1".

When you create a **path** using the **Path** wizard, the default setting displays the first **dot** when the project is loaded, and the second **dot** when the timeline reaches the end. That is, the default **GLUE** for the **path** is something like: "Attention VisualEyes! When the timeline is at the very beginning, display the first **dot**, and begin drawing the **path** line. When it reaches the very end, complete the **path** line and display the second **dot**."

This instruction is translated into language VisualEyes can understand through the *time* attribute. When defining a **dot**, giving the *time* attribute the value of "0" (zero) will display the **dot** when the timeline is at the beginning. Giving the *time* attribute the value of "1" will display a **dot** when the timeline reaches the end.

Understanding the Date Attribute

You can make the **path** more useful for displaying time-dependent data by linking the **dots** to specific dates, rather than to the beginning and end of the timeline. You do this by using the *date* attribute, which links the **dot** to a specific date on the timeline.

First, remove the *time* attribute.

- Click on the Greenwich **dot** in the Element tree.
- Click on *time*, then on Remove attribute. The attribute is removed.
- Click on the Cape May **dot**, then on the *time* attribute.
- Click on Remove attribute to remove it.

Next, add the date attribute.

- Click on the Greenwich **dot**.
- Click on Add new attribute, then on *date*.

The values for the date attribute should follow the same format as you used for the timeline. The Greenwich Tea Party occurred on December 22, 1774, so you enter the date as 12/22/1774.

- Click on the Value column for *date*, and enter "12/22/1774".

The Cape May skirmish, in which Richard Wickes was killed, thus becoming the first casualty on New Jersey soil, took place on June 29, 1776, so you can add that date to the Cape May **dot's** attributes.

- Click on the Cape May **dot**.
- Add the attribute *date*.
- Click on the Value column for *date*, and enter 6/29/1776.
- Save & Preview
- Click the Play button. The **path** should display correctly.
- Click on Return to Editing.

Using Copy and Paste in VisEdit

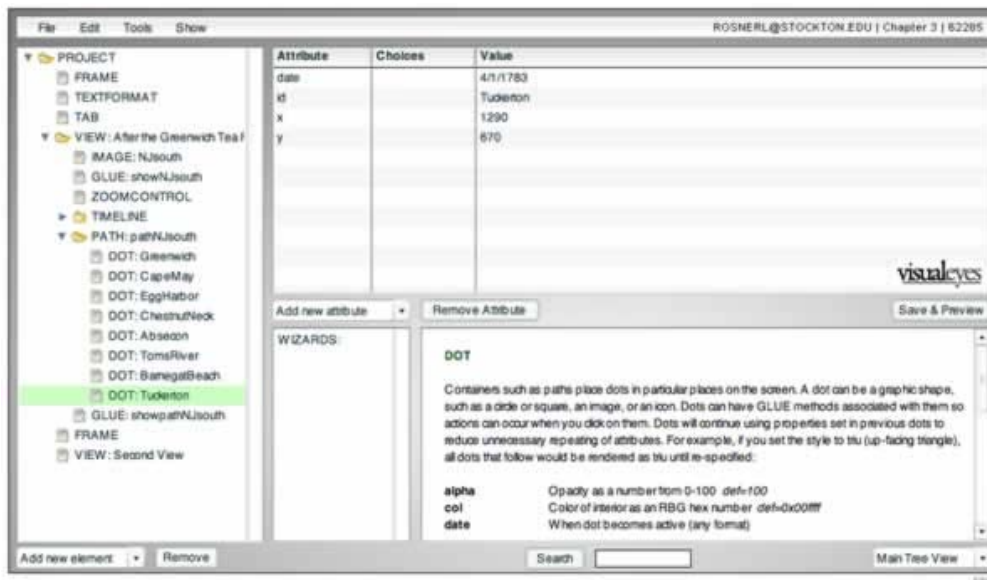
So far, so good. You have defined the **path** for New Jersey battles and defined the first two stops along the way. But how can you add more?

The easiest way to add more **dots** is simply to copy and paste the one you have. Once the first **dot** is defined, all subsequent **dots** have the same attributes unless you choose to change them. You can make copies of your second (Cape May) **dot**, then edit the *id* and coordinates for each.

For this project, you need a total of 8 **dots** to display the most memorable battles and skirmishes of southern New Jersey. You'll do this by first, creating 6 copies of the Cape May **dot**, and then modifying them to correspond to the actual locations of the hostilities.

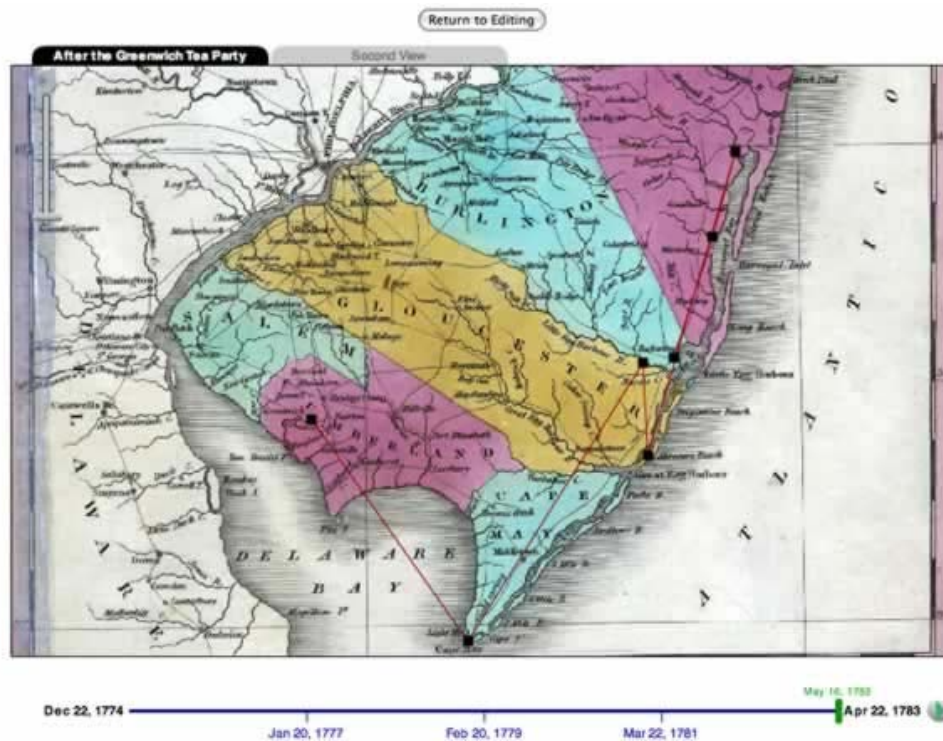
- Click on the Cape May **dot** in the Element panel.
- Click on Edit from the menu bar, then on Copy.
- Click on Edit, then on Paste. The **dot** is copied.
- Follow the same steps until you have a total of 8 **dots**, one for Greenwich and 7 for Cape May.
- Click on the third **dot** (the second Cape May **dot**).
- Edit the attributes so that the *id*= "EggHarbor", *x*= "1210" and *y*= "900", *date*= "4/26/1777".
- Edit the rest of the **dots** with the data from the following table:

<i>id</i>	<i>x</i>	<i>y</i>	<i>date</i>
Greenwich (already created)	585	790	12/22/1774
CapeMay (already created)	890	1220	6/29/1776
EggHarbor (already created)	1210	900	4/26/1777
ChestnutNeck	1230	680	10/6/1778
Absecon	1240	860	10/1/1779
TomsRiver	1410	270	5/14/1780
BarnegatBeach	1365	435	10/26/1782
Tuckerton	1290	670	4/3/1783



When the **dots** are created, you are ready to view the project.

- Click on Save & Preview, and then on the Play button. The **path** should follow the military action in southern New Jersey during the Revolutionary War. VisualEyes is a particularly effective tool for displaying that most of the engagements took place along the Atlantic coast.



- Click on Return to Editing.

Congratulations! You have successfully created and edited your **path**.

Chapter Four

In this chapter, you will learn how to enhance the ways viewers can interact with your maps and data. You will learn how to add a magnifying glass to zoom into your map, and how to add a Control Panel to your view. You will also learn advanced techniques for working with data files created with spreadsheets.

You will learn how to:

1. Understand widgets
2. Add a Magnifier
3. Add a Control Panel
4. Add an element to the Control Panel
5. Modify the Control Panel

6. Link a **path** to the Control Panel
7. Work with external data files
8. Understand data files
9. Import external data files into VisualEyes
10. Add an external data file to a **view**
11. Fill a **path** using **Dotfill**
12. Create a script from scratch

Understanding Widgets

VisualEyes allows you to add a number of *widgets* to your project. A widget is specialized tool that performs a particular task. They are highly visual and interactive, and can make the project more interesting for the user.

To see how they work, you will modify your second view to display a map of the entire length of New Jersey for the Revolutionary War period. You will then create a widget called a Magnifier, which can be dragged across the map.

As usual, start out by loading VisEdit for your project.

- Point your browser to the VisEdit screen and log in.
- Click on File, then Save as..., and save the project with the name Chapter 4.

Then, give your Second View a new title, New Jersey Battles 1775-1783.

- In the Element tree, click on VIEW: Second View to select it
- In the Attributes editor, change the *title* attribute to New Jersey Battles 1775-1783.
- Click on Save & Preview to view the new tab.
- Click on the New Jersey Battles tab to make it active.



- Then, click on Return to Editing.

The next step is to add the map. For this view, you'll use the full map of New Jersey, located at <http://www.viseyes.org/sampler/NJfull.jpg>.

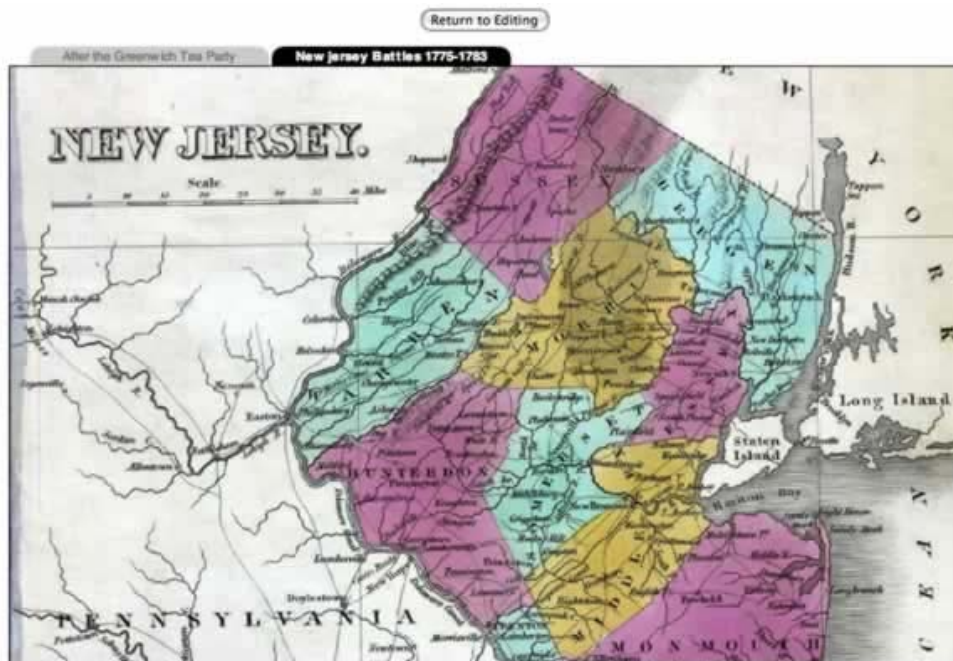
- Click on VIEW: New Jersey Battles 1775-1783 in the Element tree.

- From the Wizards panel, click on Image to start the Image wizard.

Follow the steps for the Image wizard, using the following specifications:

- For the *id*, enter "NJfull".
- For the URL, enter "<http://www.viseyes.org/sampler/NJfull.jpg>"
- For the **GLUE** *id* that will cause the map to be displayed, enter "showNJfull"
- Accept all of the rest of the default settings.
- At Step 7, click on Add, to add the map. It will appear under the **view** in the Element tree as IMAGE:NJfull.
- Click on the arrow to the left of the New Jersey Battles **view** to see its new sub-elements, **IMAGE** and **GLUE**.
- Click on Save & Preview to see your project.
- Click on the All New Jersey tab to see the map.

When the view loads, the northern half of the state of New Jersey is displayed.



You can drag the map to see the southern part of the state.

- Practice dragging the map to see the northern and southern sections. When done, click Return to Editing.

Adding a Magnifier

A Magnifier is a visual tool, shaped like a magnifying glass, for zooming into a section of an image. It creates a window from the original image through to any image you specify, generally a zoomed-in version of the original. It makes working with images more interesting and interactive.

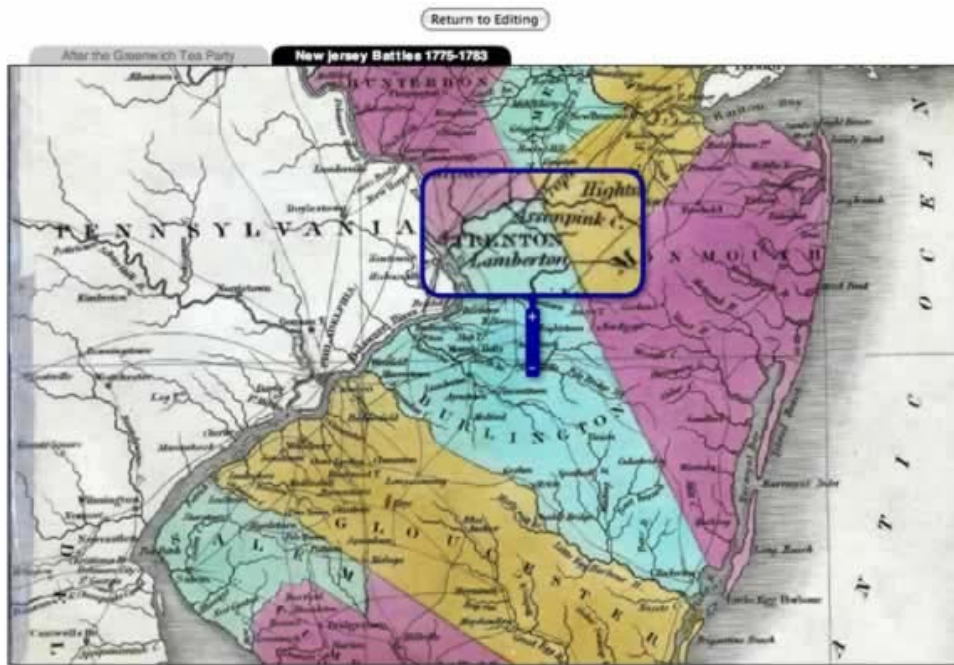
The easiest way to add a magnifier is to use the wizard.

- Click on VIEW: New Jersey Battles 1775-1783 if it is not already highlighted.
- From the Wizards panel, scroll down until you see Magnifier.
- Click on Magnifier to begin the wizard.
- Accept all default settings to step three when you are prompted for the URL of the image. In this case, you want to magnify the image you are already using, so enter <http://www.viseyes.org/sampler/NJfull.jpg> and click on Next step.
- Continue the wizard, accepting the default settings.
- When the Done prompt appears, click on the Add button. The Magnifier widget will be added.
- Click on Save & Preview, then on the New Jersey Battles tab. The Magnifier appears.

If you drag the Magnifier on the map, though, you'll notice something is wrong. It doesn't seem to be magnifying anything!

The reason it doesn't appear to be working is that the default magnification is set to 1, that is, no magnification at all. To change that, click on the + sign on the Magnifier handle. Now it should work properly!

- Practice dragging the Magnifier around the map, and using the + and - signs to change the level of magnification.



- Click on Return to Editing.

Adding a Control Panel

For the next part of the project, we will add a **path** with **dots** representing New Jersey battles. We would like to use the same map of the entire state. One option is to create a third **view** and use the same map for the background. There is, though, a more efficient way to use our resources: we can create a Control Panel to organize our material, so that we can present different types of information within the same **view**.

You can create a Control Panel with a wizard:

- Click on VIEW:New Jersey Battles 1775-1783 if it is not already highlighted.
- From the Wizard panel, click on the Control Panel wizard.
- Follow the steps of the wizard in the usual way, clicking on Next Step to select the defaults.
- Specify the *title* as "New Jersey Map".
- At the Done prompt, click the Add button. The Control Panel is created.
- Click on Save & Preview, and on the New Jersey Battles tab to see it.



Your Control Panel has been created, but it does not yet do anything. Now we'll add the attributes to make it work.

- Click on Return to Editing to return to the Main Tree View, and click on CONTROLPANEL in the Element tree to show its related elements and attributes.

The Control Panel wizard creates the structure for the Control Panel, but you have to add **elements** to create the actual controls. For this project, you will create two check boxes. These boxes work as simple on/off switches. When checked, they are on, and they display the specified information. When unchecked, they hide the specified information.

Adding an Element to a Control Panel

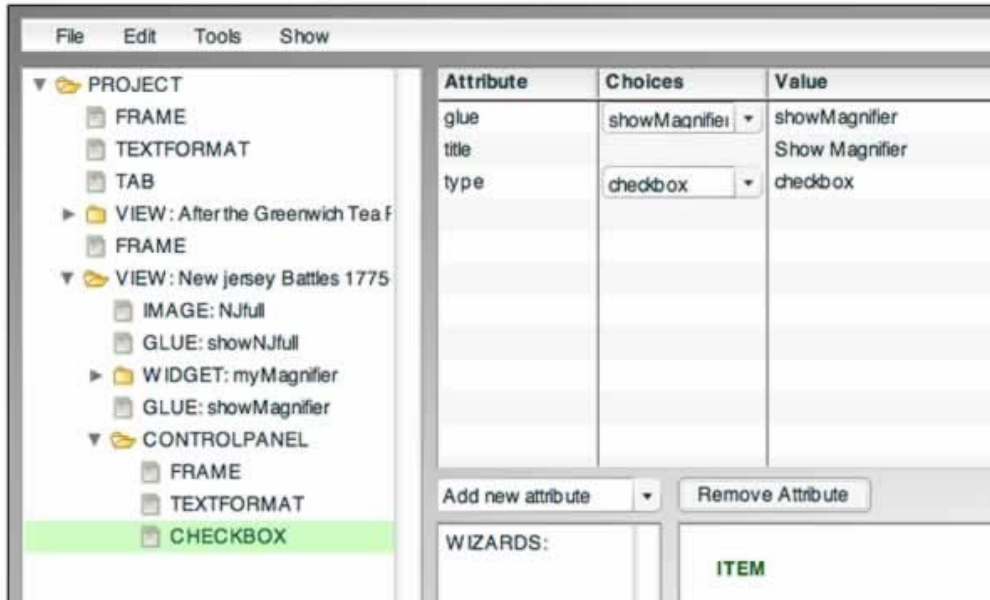
The first **element** you add will control whether or not the Magnifier is displayed:

- Make sure CONTROLPANEL is highlighted in the Element tree, and click on the "Add new element" menu to display the options.
- Choose **item** from the menu.
- In the Attribute editor, click on *title*, and enter "Show Magnifier".
- Click on *type*, and select "checkbox" from the menu.

The two attributes you just added control the appearance of the element. It will be displayed as a check box, with the title Show Magnifier. The next attribute you add will control what the check box actually does. It is a *glue* attribute, which will run a specified script. You've already created

the **GLUE** you want to run: showMagnifier, the script that tells VisualEyes to display the Magnifier.

- From the “Add new” list box in the Attribute editor, select *glue*. The *glue* attribute provides a menu with all the *glue* scripts you have created to date.
- Select “showMagnifier”.

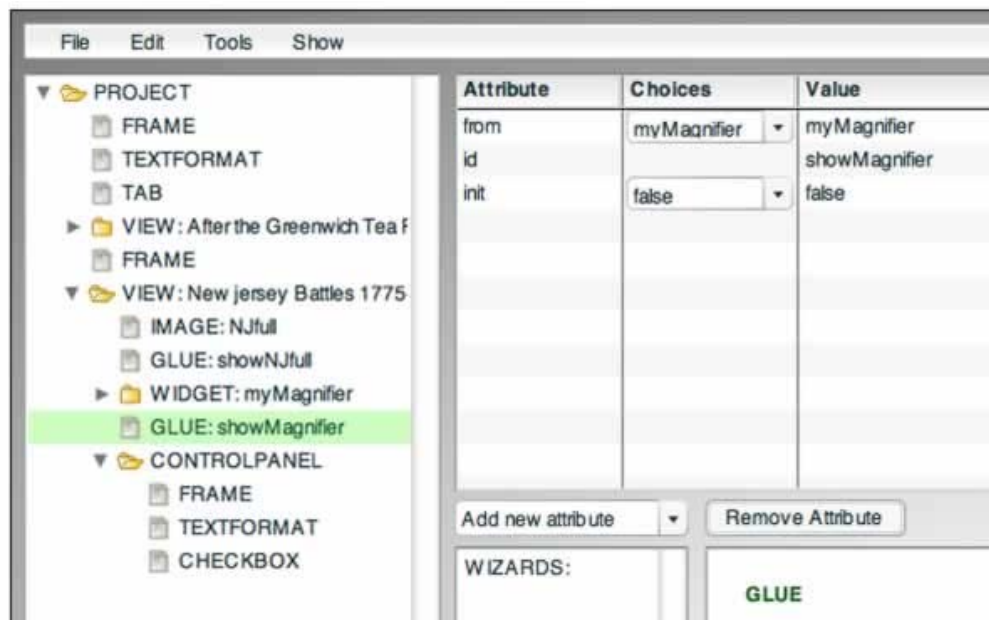


The *glue* attribute you have just added tells VisualEyes to display the Magnifier when the check box in the Control Panel is checked. For it to work properly, you will have to adjust the showMagnifier **GLUE** slightly.

- From the Element tree, click on GLUE:showMagnifier to display its attributes.

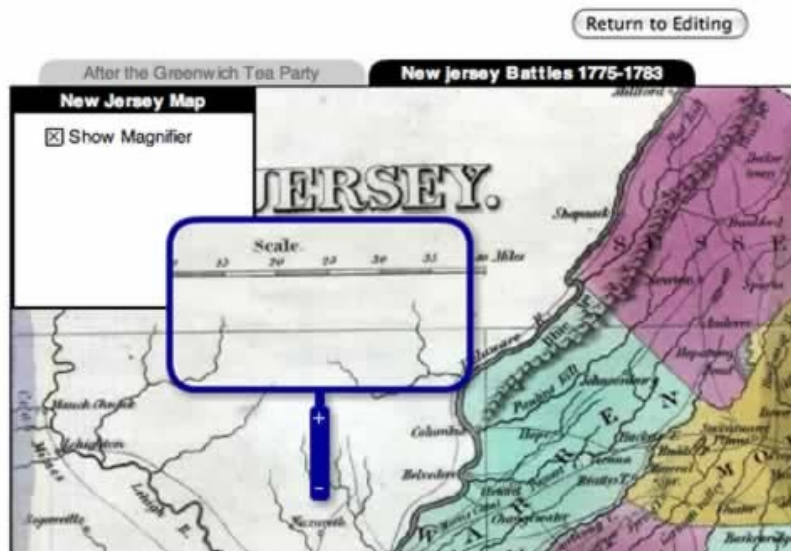
In the Attributes editor, you can see that *init* is set to "true". That means that the **GLUE** will automatically run when the project is loaded. You want to modify it so that it will only run when the check box in the Control Panel is checked.

- Click on *init* in the Attribute panel, then on the menu in the Choices column.
- Select "false".



Now the "showMagnifier" **GLUE** will not run automatically. Instead, it will only run when the **GLUE** is called, that is, when the **checkbox** in the Control Panel is checked.

- Click on Save & Preview, and then on the New Jersey Battles tab. As you can see, the Magnifier no longer appears in the view.
- Click on the Show Magnifier check box in the Control Panel. The Magnifier appears.



Modifying the Control Panel

The Control Panel does what it is supposed to, but it could look a little bit prettier. Why not change the opacity, so that you can see the map beneath?

- Click on Return to Editing, then on FRAME under CONTROLPANEL in the Element tree.

The attribute that controls the opacity of the Control Panel is *alpha*. When it is set to "100", the feature is fully opaque. Try changing it to "75".

- Click on the *alpha* attribute, and change the value to "75".
- Click on Save & Preview, then on the New Jersey Battles tab to see the change. The Control Panel is partially transparent.

Comment [1]: Didn't work for me, the Control Panel is still fully opaque. It is still opaque when I set alpha to 50. —mvp5a



There are other ways you can modify a Control Panel. You can adjust the appearance of the Control Panel by changing the color of the background and text, and its height and width. You can also adjust where it appears in the view. You can see the attributes that control its overall appearance by selecting FRAME in the Element tree. You can see the attributes that control the font size and appearance by selecting TEXTFORMAT in the Element tree.

Congratulations! You have successfully added a widget to your project!

- Click on Return to Editing.

This has been a complicated section. It's a good time to save your work to make sure you don't lose it.

- Click on File, then on Save to save your work.

Adding a *Path* to the Control Panel

We can now move on to the next part of our project: adding a *path* of Revolutionary War battles in New Jersey. You already know the basic steps for adding a path. You first create it and define its attributes, then add *dots* and define their attributes. Finally, you create a **GLUE** to show the path. You can have the *path* display when you click a checkbox on the Control Panel by linking to its **GLUE**, just as you did for the Magnifier.

Begin by defining the *path* itself:

- Click on VIEW:New Jersey Battles 1775-1783 if it is not already highlighted.
- From the Wizards list, select Path.

Follow the steps for the Path wizard.

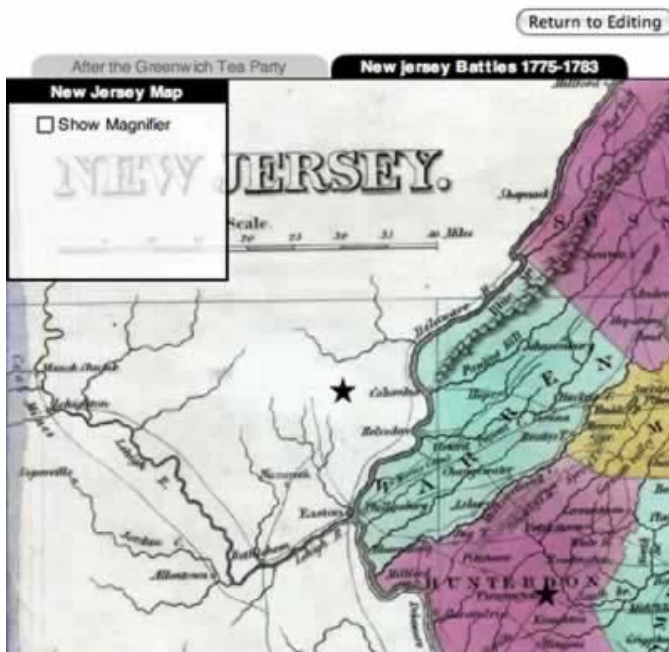
- Give the **path** the *id* "pathBattles" (step 2).
- Keep the default for the line color (step 3).
- When prompted for the width (*wid*), enter "0" (step 4).
- When prompted for Tween lines between **dots**, select "false" (step 5). That means that you do not want lines to be drawn between the **dots**.
- Keep the default for the first **dot** (step 6). You'll change it later.
- Select "star" for the **dot** style (step 7).
- Enter "40" when prompted for the **dot** size (step 8).
- Keep the defaults for the second **dot** (step 9) and adding **GLUE** to show the path (step 10).
- Give the **GLUE** the *id* "showpathBattles" (step 11).



- Click "Add" to finish the wizard (step 12) and return to the Main Tree View.

You'll want to preview the **path** to make sure it's working right. But first, make a few changes to make the **dots** easier to see:

- Click on the first **dot** in the Element tree to select it (you may have to click on the arrows next to its VIEW and PATH), and edit its attributes so that x= "500" and y= "500".
- Remove the *time* attribute. The *time* attribute links the **dot** to a specific point on the timeline. Since you are not working with a timeline, you don't need it right now.
- Click on the second **dot** to select it, and edit its attributes so that x= "800" and y= "800".
- Remove the *time* attribute.
- Save & Preview, and click on the New Jersey Battles tab. The two **dots** should appear as you specified them.



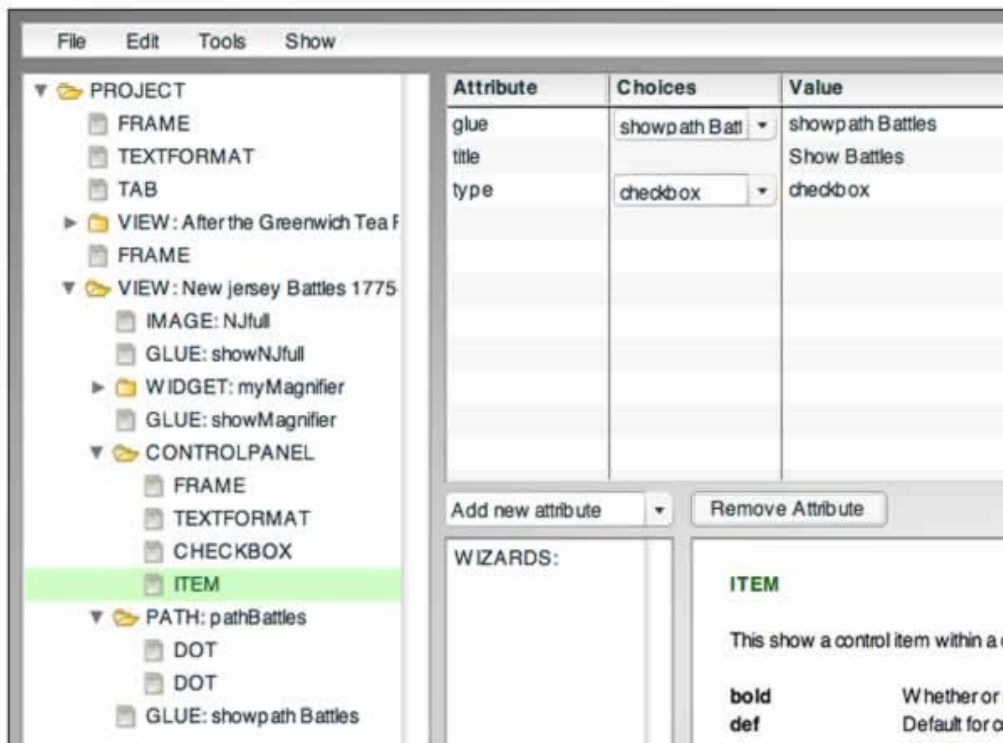
- Return to Editing.

The next step is to edit the path's **GLUE** so that you can call it from the Control Panel checkbox:

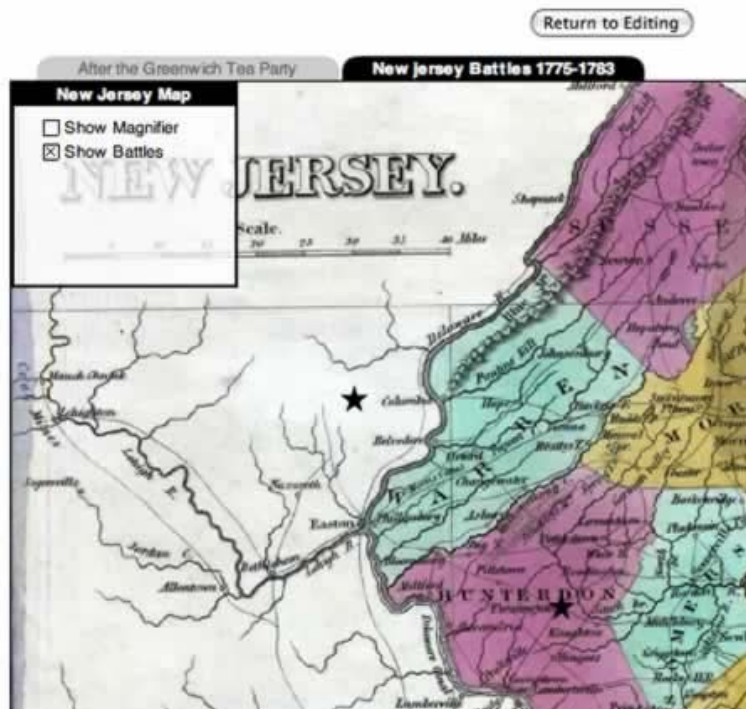
- In the Element tree, click on GLUE:showpathBattles to select it.
- Select *init* from the Attribute tree, and select "false". That means that the **path** won't show up when the project is initially loaded. Instead, it will wait until it is called – in this case, by checking on a checkbox in the Control Panel.

You have now set up the **PATH** "pathBattles", and the **GLUE** "showpathBattles" that will allow that **path** to be displayed. The next step is to create a second checkbox on the Control Panel.

- Click on CONTROL PANEL in the Element tree to select it, and click on the "Add new element" menu to display the options.
- Choose **item** from the menu.
- In the Attribute editor, click on *title*, and enter "Show Battles".
- In the Attribute editor, select "checkbox" for the *type*.
- Select *glue* from the Add new attribute menu, and choose "showpathBattles".



- Save & Preview, and select the New Jersey Battles tab.
- Click on the Show Battles checkbox. The **path** will appear.



- Return to Editing.

You have used the **dots** to make sure that your **path** displays properly, but you won't need them again.

- Remove the two **DOTs** from the Element tree. Be sure not to remove the **PATH** or **GLUE**!

Working with External Data Files

Last time you created a path, you added each **dot** individually. This time, we will use an *external data file*, together with the *filldot* script, to add the **dots** to the path. An external data file can be created in another program, like an Excel spreadsheet. This can be very helpful for projects in which an assistant enters the data. In order to make the best use of this feature, you'll find it helpful to understand how VisualEyes processes data files.

Understanding Data Files

In many software programs, data is organized into units, often called *records*. Each record contains certain pieces of information, called *variables*. A very familiar kind of data organized in this way is an address book. Each person in the address book is a separate record. Each piece of information about the person – name, address, cell number – is a variable. In many programs, including Excel spreadsheets, records and variables are organized into tables of rows and columns, as in the example below:

Name	Address	Cell
Smith, John	104 East Lansing St, Aberdeen, MD	401-345-8726
Roberts, Mary	43 South Omaha Ct, Briggs, CA	323-969-3871
Tucker, Bill	8 North Chestnut Ave, Mt Oliver, PA	412-123-7538
Damato, Jane	9372 West Broad St, East Troy, WI	262-456-0892

In this table, the names of the variables are "Name", "Address", and "Cell". The data is organized according to people, and each person is a separate record. Each person has distinct values for each variable, and those values are entered into the appropriate column for that variable.

When working with VisualEyes, you can organize the data for each **dot** in the same way. Each **dot** is a separate record, and **dot's** attributes are its variables. With that in mind, we can take another look at the data we entered for PATH:pathNJsouth:

<i>id</i>	<i>x</i>	<i>y</i>	<i>date</i>
Greenwich	585	790	12/22/1774
CapeMay	890	1220	6/29/1776
EggHarbor	1210	900	4/26/1777
ChestnutNeck	1230	680	10/6/1778
Absecon	1240	860	10/1/1779
TomsRiver	1410	270	5/14/1780
BarnegatBeach	1365	435	10/26/1782

Tuckerton	1290	670	4/3/1783
-----------	------	-----	----------

When you created each **dot**, you created a record, like a row in an Excel spreadsheet. Each *attribute* – *id*, *x*, *y*, and *date* – is a variable, and the "value" you entered for each – for example, "Greenwich", "CapeMay", or "EggHarbor" for *id*, and "585", "890", or "1200" for *x* – provides information to VisualEyes on how to define the **dot** and locate it in time and space.

For the **path** you've just created, pathBattles, you will be adding the following **dots**. Take a few moments to look at the data set in the table

<i>id</i>	<i>lab</i>	<i>x</i>	<i>y</i>	<i>style</i>	<i>wid</i>	<i>col</i>	<i>glue</i>	<i>year</i>	<i>mark</i>
Absecon	Absecon	1208	1885	star	40	0x660000	showBattles	1779	0
Amboy	Amboy	1200	829	star	40	0x660000	showBattles	1776	1
Barnegat	Barnegat	1325	1490	star	40	0x660000	showBattles	1782	2
Bordentown	Bordentown	925	1160	star	40	0x660000	showBattles	1778	3
Brigantine	Brigantine	1240	1798	star	40	0x660000	showBattles	1775	4
CapeMay	Cape May	865	2205	star	40	0x660000	showBattles	1776	5
Camden	Camden	620	1320	star	40	0x660000	showBattles	1778	6
Cranberry	Cranberry	1093	988	star	40	0x660000	showBattles	1777	7
DelawareBay	Delaware Bay	585	2090	star	40	0x660000	showBattles	1780	8
EggHarbor	Egg Harbor	1070	1915	star	40	0x660000	showBattles	1779	9
Elizabethtown	Elizabethtown	1295	670	star	40	0x660000	showBattles	1779	10
Hackensack	Hackensack	1400	480	star	40	0x660000	showBattles	1777	11
Haddonfield	Haddonfield	645	1385	star	40	0x660000	showBattles	1778	12
Manasquan	Manasquan	1430	1220	star	40	0x660000	showBattles	1780	13
Monmouth	Monmouth	1260	1000	star	40	0x660000	showBattles	1778	14
Morristown	Morristown	1128	550	star	40	0x660000	showBattles	1777	15
MtHolly	Mt. Holly	857	1270	star	40	0x660000	showBattles	1776	16
NewBrunswick	New Brunswick	1148	835	star	40	0x660000	showBattles	1777	17
Newark	Newark	1330	600	star	40	0x660000	showBattles	1780	18
Paramus	Paramus	1370	370	star	40	0x660000	showBattles	1780	19
PaulusHook	Paulus Hook	1350	655	star	40	0x660000	showBattles	1779	20

Princeton	Princeton	960	960	star	40	0x660000	showBattles	1777	21
RedBank	Red Bank	1417	980	star	40	0x660000	showBattles	1780	22
SandyHook	Sandy Hook	1450	870	star	40	0x660000	showBattles	1782	23
Shrewsbury	Shrewsbury	1155	1135	star	40	0x660000	showBattles	1779	24
Somerset	Somerset	990	760	star	40	0x660000	showBattles	1779	25
Spanktown	Spanktown	1244	730	star	40	0x660000	showBattles	1781	26
TomsRiver	Toms River	1347	1362	star	40	0x660000	showBattles	1778	27
Trenton	Trenton	880	1065	star	40	0x660000	showBattles	1776	28
Tuckerton	Tuckerton	1250	1693	star	40	0x660000	showBattles	1783	29
Woodbridge	Woodbridge	1245	765	star	40	0x660000	showBattles	1780	30

The information for each **dot** is displayed in a separate row, with the value for each variable in a separate column. Most of the variables provide VisualEyes with the attributes it needs to define each **dot**, and you have seen many of them before. These are:

<i>id</i>	The id for the dot
<i>x</i>	Location of the dot on the map, using the x,y coordinates
<i>y</i>	Location of the dot on the map, using the x,y coordinates
<i>style</i>	Symbol used to define the dot . In the data table above, the value is "star"
<i>wid</i>	Width of the dot in pixels. In the data set above, the value is "40"
<i>col</i>	Color of the dot . In the data set above, the value is "0x660000"

The data table also includes some new attributes for you to work with:

<i>lab</i>	Label for the dot . You can use this label to identify the dot to the viewer. It can be the same as, or different from, the <i>id</i> .
------------	---

<i>glue</i>	This <i>glue</i> allows you to link each dot to a VisualEyes feature or component. When the viewer interacts with the dot – by clicking on it, for example – the glue will "call" another part of the program. You'll use the <i>glue</i> attribute to connect each dot to an infoBox in the next chapter.
-------------	---

The final two variables in the data table, *year* and *mark*, have no special meaning to VisualEyes. Instead, they add additional information about each record, which will be helpful in displaying the data. You'll use them to display information in infoBoxes in the next chapter.

<i>year</i>	The year that the battle took place. Note that we are NOT calling this variable <i>date</i> . That's because the attribute <i>date</i> has special meaning in VisualEyes: it connects each dot to a point on the timeline. Since we're not using a timeline for this part of the project, we don't want to confuse VisualEyes by creating a <i>date</i> attribute.
<i>mark</i>	A unique number that belongs to one, and only one, record. We'll discuss its purpose in the next chapter.

Importing Data into VisualEyes

VisualEyes allows you to import data you have created using Excel or other software programs. The data can be uploaded to the server and converted to a file format VisualEyes can read. You specify the data file by pointing to its URL as an attribute, as you would for image files. The commands for this are located on the Tools menu.

For this tutorial, the data has been uploaded for you. It is located at <http://www.viseyes.org/sampler/NJbattlesdata.csv>.

Adding an External Data File

Now that you understand the data, it is time to add it to the New Jersey Battles **view** as a **resource**.

- Click on VIEW:New Jersey Battles 1775-1783 to select it.
- Click on “Add new element” in the Element tree, and select Resource from the menu.
- In the Attribute editor, enter the *id* for the **resource** as "dataBattles"
- For *type*, select "xml"
- From the Add new attribute menu, select *src*. Enter the location for your data, ["http://www.viseyes.org/sampler/NJbattlesdata.csv"](http://www.viseyes.org/sampler/NJbattlesdata.csv)
- From the Add new attribute menu, select *preload*, and select "true". This ensures that the data loads when you load the view.

Filling a *Path* with a Data File

You have added the New Jersey battles data to the project. The next step is to use the data to fill up the *path* with **dots**. To understand how to do that, imagine that you have a research assistant working for you, and that you would like that research assistant to create each **dot** individually, as you did for the PATH:pathNJsouth. You might say give him/her instructions like: "Please take the information for each of the **dots** contained in this data source, and create the **dots** within this path. Please add each of the **dots** in the order in which they appear in the original data set."

These instructions contain three important pieces of information:

1. What action you want taken (fill a *path* with **dots** from a data source)
2. Where the information for the **dots** should come from (the specified data source)
3. Where the **dots** should go (into the specified path).

From these instructions, your research assistant would know exactly how you wanted to create the **dots** for the path.

You can give the same instructions to VisualEyes, as long as you use the right language. The language you use is a *script*, contained within a **GLUE** element. If you could speak to VisualEyes in English, you might say: "Please take the information for each of these **dots** from the specified data set (the **resource** with the *id* "dataBattles"), and add it to the specified *path* (the *path* with the *id* "pathBattles". To speak to VisualEyes in a language the program can understand, you make use of the **dotfill** command. To use **dotfill**, all you need is the *id* of the *path*, and the *id* of the data **resource** you are using to create the **dots**. In this case, the *path id* is "pathBattles", and the data **resource id** is "dataBattles". The script will therefore look like this:

```
dotfill(pathBattles,dataBattles)
```

1. **dotfill** tells VisualEyes what you want done (create **dots** from a data source)
2. **pathBattles** tells VisualEyes where the information for the **dots** should go (the specified path)

3. **dataBattles** tells VisualEyes where the information for the **dots** should come from (the specified data source)

Please note that there should not be any spaces in the script before or after parentheses or commas.

Creating a Script from Scratch

In previous sections, you have created **GLUE** elements and scripts as part of wizards. In this case, you will create a **GLUE**, and its script, from scratch. This is precisely the feature that makes VisualEyes such a flexible software tool: it allows users to write small scripts for specific commands and attach each script to its own **GLUE**. That **GLUE** can then be called in a variety of ways, for example at startup, from a checkbox, or from a timeline.

In this case, we will create a **GLUE** to load the **dots** into our **path**, "pathBattles". Start by creating the new **GLUE** element:

- Click on VIEW:New Jersey Battles 1775-1783 if it is not already highlighted.
- From the Add new element menu, select **glue**.
- From the Attribute editor, click on Add new attribute, and select *id*.
- Give the **GLUE** the *id* "fillBattles".
- Add the attributes *init* and *once*, and set them both to "true". This will direct VisualEyes to fill the **path** with the specified data only once, when the data is loaded.

Next, you will add the specific commands.

- Click on Add new attribute, and select *script*.
- Delete "Insert script code here" from the lower panel, and insert the *script* "**dotfill**(pathBattles,dataBattles)".

Do not include the quotation marks or formatting, and remember that there must not be any blank spaces between characters.

Congratulations! You have successfully used an external data file to add **dots** to a **path**, and you have controlled the display of the **path** with the Control Panel.

Chapter Five

In this chapter, you will learn how to create an infoBox that appears whenever a **dot** is clicked. The infoBox will display the location and year of the battle associated with each **dot**. You will also learn more about how VisualEyes links data, as well as more about working with scripts.

You will learn how to:

1. Create a simple infoBox attached to a **dot**
2. Display variables in an infoBox
3. Use a script with an infoBox
4. Use a query in a script
5. Use \$\$click to select data
6. Use status() to display data
7. Use replaceword to fill an infoBox from a list

Creating an InfoBox

An infoBox is one of the most versatile features of VisualEyes. It can be set up to appear when clicked, like a pop-up box on a webpage. For this project, we will set up an infoBox to appear when each of the New Jersey battle **dots** is clicked. The infoBox will display the location and date for each battle.

As usual, start out by loading VisEdit for your project, and saving it with a new name.

- Point your browser to the VisEdit screen and log in.
- Click on File, then Save as..., and save the project with the name Chapter 5.

You can use a wizard to create the infoBox:

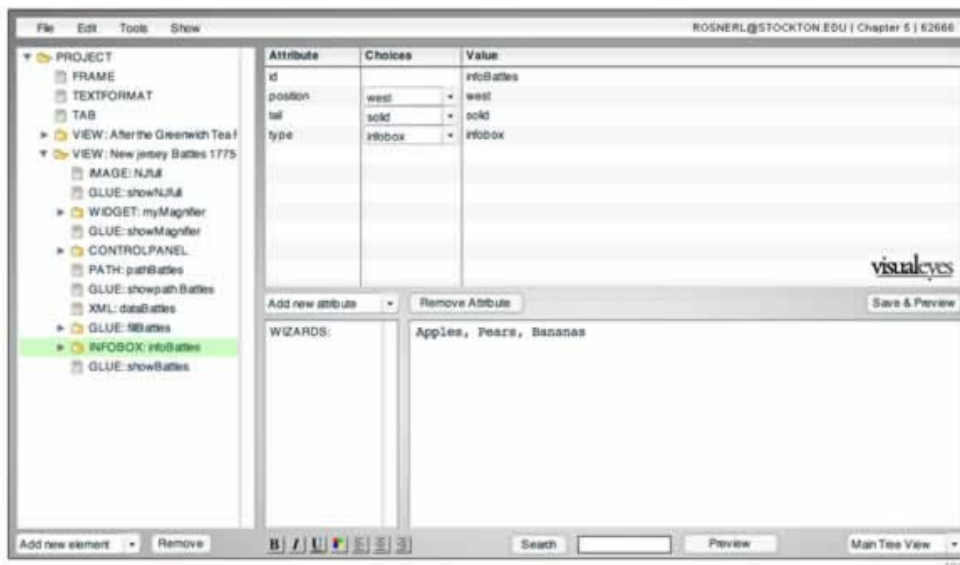
- Click on VIEW:New Jersey Battles to select it, and select Infobox from the Wizards list.

Follow the prompts for each step:

- In Step 2 give the infoBox the *id* "infoBattles"
- In Step 3 choose a *solid* tail (the tail connects the box to the **dot**) from the menu.
- In Step 4 type "Apples, Pears, Bananas" in the *text* to display in the box.

You're just entering this text as a placeholder, to make sure the infoBox displays properly. You'll change it later.

- In Step 5 select "west" from the pull-down menu for *position* of the box.
- In Step 7 give the infoBox a FRAME *width* of 175 pixels.
- In Step 8 give the infoBox a FRAME *height* of 75 pixels.
- Accept the default settings for Steps 9-11.
- In Step 12 change the **GLUE** *id* to "showBattles".
- Click Add to finish the wizard.
- Click on INFOBOX:infoBattles in the Element tree to see what you've added. (you may have to click on the arrow next to VIEW:New Jersey Battles 1775-1783 to display it).



You can see the specifications you've created for your infoBox. The attributes specify how the box will look. The text "Apples, Pears, Bananas" is what will appear in the box.

You can also see the **GLUE** you've created to display the infoBox, showBattles. You may remember that is the name of the *glue* attribute included in the dataBattles file. By giving the **GLUE** to display the infoBox that *id*, you are giving a command to VisualEyes: You are saying, "Whenever any of the **dots** in PATH:pathBattles is clicked, display INFOBOX:infoBattles".

You can see how it works by viewing your project.

- Click on Save & Preview, and on the New Jersey Battles tab.
- Click on Show Battles in the Control Panel to display the **path**.
- Click on any of the **dots**. The infoBox is displayed.



The infobox as you have set it up will appear when any of the **dots** are displayed. If you want to change the background color or the size of the box, you can change the attributes of its **frame**. It will always display the same text, "Apples, Pears, Bananas", because that is what we specified in the wizard.

- Click Return to Editing to return to the Main Tree View.

Displaying Variables in an Infobox

Of course, we don't want the infoBox to display a list of fruit. It's worthwhile thinking at this point exactly what we do want. We want a viewer to be able to click on each **dot**, and have the infoBox display the location and year of the battle associated with that **dot**. It will make it easier to work with VisualEyes if we separate this task into a number of steps:

1. We click on a **dot**, which calls a **GLUE** to show the infoBox. We've already created this **GLUE**, showBattles.
2. We want VisualEyes to go through our list of **dots** in the data file (in the Element tree as XML:dataBattles), and find the data associated with the specific **dot** we clicked.
3. We want VisualEyes to look in that data to find the location of the **dot** and the year. We included the information for location as the variable *lab* (short for label) and the information for the year as the variable *year*.

- Once VisualEyes has found that information, we want it displayed to as the text for the infoBox. In other words, instead of "Apples, Pears, Bananas", we want VisualEyes to display the text for each **dot**, for example, "Trenton, 1776".
- And we want VisualEyes to do exactly the same thing for each **dot** we click, each time locating the data associated with that **dot** (and only that **dot**). That is, "Trenton, 1776" should be displayed when we click the **dot** located at Trenton, while "Princeton, 1777" should be displayed when we click the **dot** located at Princeton.

One way to do this would be to create a separate infoBox for each **dot**, and a separate **GLUE** to call each infoBox. But that would be a lot of work: for our New Jersey Battles data, we would have to create 30 infoBoxes and **GLUE** elements, and enter the data for each. A more efficient way of doing it would be to create just one infoBox, and ask VisualEyes to change the information in it, depending on which **dot** is clicked.

Using a Script with an Infobox

We can ask VisualEyes to do all these things by including a *script* in the **GLUE** that displays the infoBox, GLUE:showBattles.

- Click on GLUE:showBattles in the Element tree to select it.
- In the Attribute editor, add the attribute *script*.
- Enter the first line of the script exactly as shown below (no spaces between parentheses or commas, and be sure to include the final comma before the close parenthesis):

query(\$battles,dataBattles,lab+year+mark,mark EQ \$\$click,)

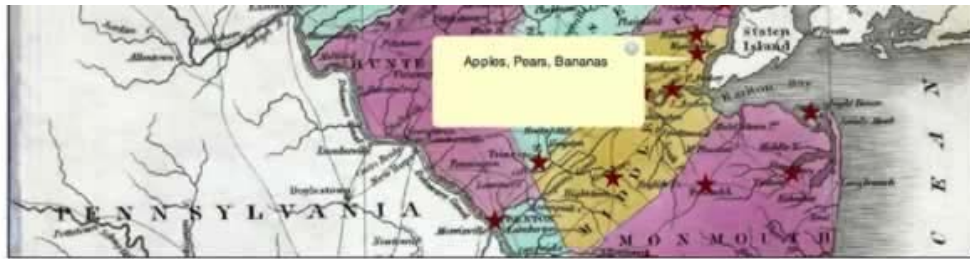
- Then, enter the second line exactly as shown (again, no spaces):

status(\$battles)



- Click on Save & Preview, then go to the New Jersey Battles tab.
- Click on Show Battles in the Control Panel, and on any of the **dots**.

The infoBox itself will look the same, since you have not changed its appearance. But if you look down at the lower left corner of the screen, you'll see three pieces of information: the location, year, and mark associated with each **dot**.



VisualEyes (1780.0)

18/11

- Click on other **dots**, checking the lower left corner after each click. The information changes to correspond to each **dot** clicked.

How did that happen? Let's find out.

- Click on Return to Editing
- Click on GLUE:showBattles to take a closer look at the script.

Using a Query in a Script

The first line of script sets up a *query* to select the data. A query is a question you are asking your data. In VisualEyes, you can set up a query and attach it to a **GLUE**. Like other scripts, the query must follow a specific format for VisualEyes to be able to understand it.

The command you have entered, "query(\$battles,dataBattles,lab+year+mark,mark EQ \$\$click,)" gives VisualEyes a set of instructions, going in order from from left to right. We will work through the instructions step by step.

1. The first step,
query(\$battles,

creates a new *list* called \$battles. A list is a container to put data in. In this step, the query command tells VisualEyes to create the list and give it the name \$battles. So far, there is nothing in it.

2. The second step,
`query($battles,databattles,`

tells VisualEyes that the data for the new list \$battles will come from information in the data **resource** used to define each **dot**, XML:dataBattles

3. The third step,
`query($battles,dataBattles,lab+year+mark,`

tells VisualEyes to go through all the data records in order, select the information for three of the variables: *lab*, *year*, *mark*, in that order, and put that information in the list \$battles.

If the list \$battles were printed, it would look like a series of rows, one per **dot**, with three columns, for *lab*, *year*, and *mark*. It is a subset of the data found in dataBattles, with all 30 of the records, but only three variables.

4. The fourth step
`query($battles,dataBattles,lab+year+mark,mark EQ $$click,)`

tells VisualEyes to go through all the records in dataBattles and for each **dot**, compare the value of the variable *mark* with the value of the variable VisualEyes creates to keep track of the order of **dots** in each path, called *\$\$click*. If the value of *mark* is equal to the value of the variable *\$\$click*, then display the data for that **dot**.

Using \$\$click to Select Data

VisualEyes creates the variable *\$\$click* whenever a **path** is created. The value of *\$\$click* is numbered consecutively, starting at 0 with the first **dot** in the path, and continuing on through all the **dots** in that path. The **dots** are ordered by the way they appear in the **view** (if they were entered individually) or by the way they appeared in the data file. In PATH:pathBattles, the first **dot** is Absecon, so VisualEyes assigned the value of *\$\$click* to "0". The last **dot** is Woodbridge, and VisualEyes assigned the value of *\$\$click* to "30". VisualEyes assigns each **dot** in the **path** one, and only one, value for *\$\$click*.

When we created the data file for New Jersey battles for pathBattles, we created the variable *mark* as an identification number, a unique identifier for each record. Each record has one, and only one value for *mark*. We assigned the values to go in the same order, so the value of *mark* for Absecon is "0", and the value of *mark* for Woodbridge is "30". That means that, for any given path, there can only be one value of *\$\$click* that matches any value of *mark*. It also means that, when VisualEyes follows the commands in our query, there will be one, and only one, record in the list \$battles.

Many businesses use a similar technique to match customer id, held in one database, with order information, held in another.

Using Status() to Display Data

The next line of the script, **status(\$battles)**, tells VisualEyes to display the current contents of the list named \$battles. Remember that the current contents have been set by the query, so the list \$battles will only ever contain the values for the **dot** that has been clicked. If no **dot** is clicked, the list is empty. VisualEyes is set up so that status() is always displayed in the lower left corner.

You can also use the status command to display the current value for \$\$click.

- Click on GLUE:showBattles to select it.
- Edit the script so that the status command reads: **status(\$\$click)**.



- Click on Save & Preview, and on the New Jersey Battles tab.
- Click on Show Battles in the Control Panel.
- Check the results by clicking on the New Jersey Battles **dots**. The value for \$\$click should appear in the lower left corner. You can check that the **dot** at Absecon appears as 0, and the **dot** at Woodbridge as 30.
- Return to Editing.

Using ReplaceWord to fill an Infobox from a List

- The last step for our infoBox is to display the data for each **dot** in the box. Just as we used the script fill**dots** to fill the **path** with data for each **dot**, we can use the script replaceword to fill the infoBox with the **lab** and **year** variables for each **dot**.

Once again, we can divide the necessary tasks into steps we want VisualEyes to take:

1. We want VisualEyes to get the *lab* and *year* values for each **dot** (already accomplished with the query, above).
2. We want VisualEyes to replace the text in INFOBOX:infoBattles with the *lab* and *year* values. This can be subdivided into two steps:
 - a. In INFOBOX: infoBattles, replace the text "Apples, Pears, Bananas" – which VisualEyes interprets as constant text, that is, text that should appear all the time – with *placeholders* for three words appearing in sequence. In VisualEyes, those placeholders are written as \$\$1, \$\$2, and \$\$3. VisualEyes interprets those placeholders as meaning “replace \$\$1 with the first word in the list, replace \$\$2 with the second word in the list, and replace \$\$3 with the third word in the list”.
 - b. Use the replaceword command to replace the placeholders with the data for each **dot**.

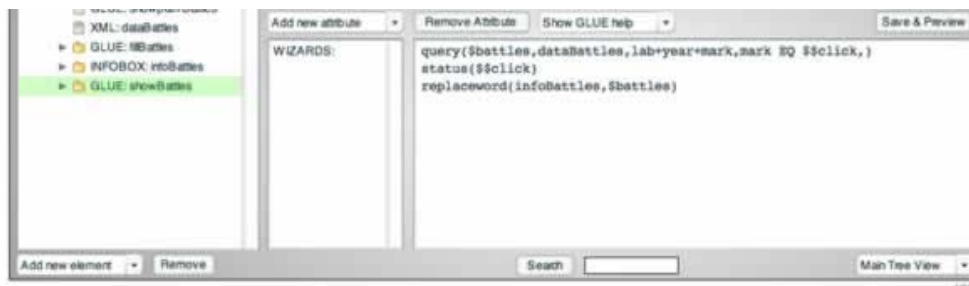
Begin by editing the infoBox:

- Click on INFOBOX:infoBattles in the Element tree to highlight it.
- Delete Apples, Pears, Bananas from the lower panel, and replace it with the placeholders: \$\$1,\$\$2,\$\$3



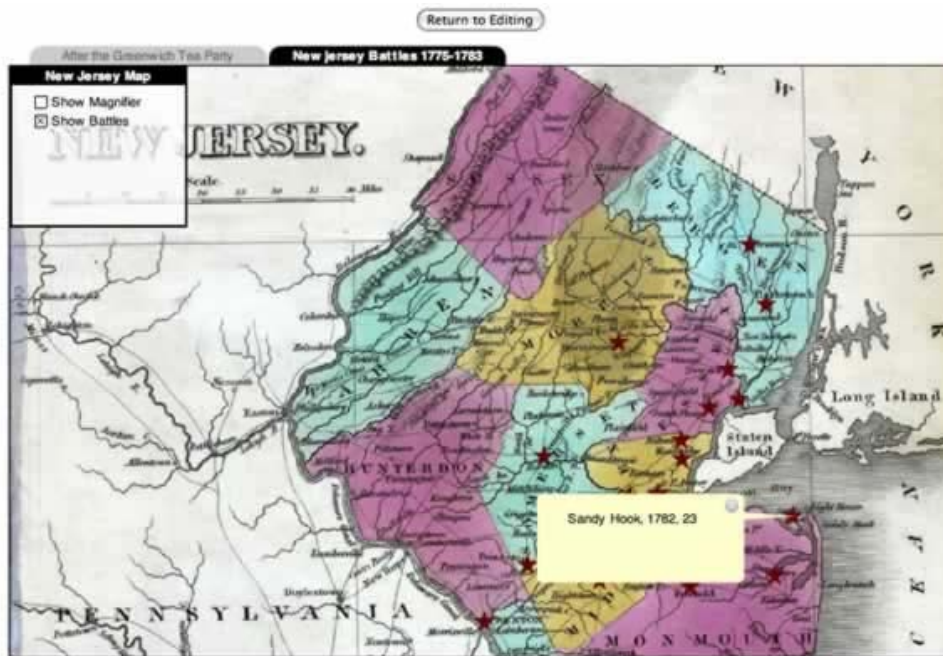
- Click on **GLUE:showBattles** in the Element tree.
- Add another line of script, typed exactly as below, with no spaces:

replaceword(infoBattles,\$battles)



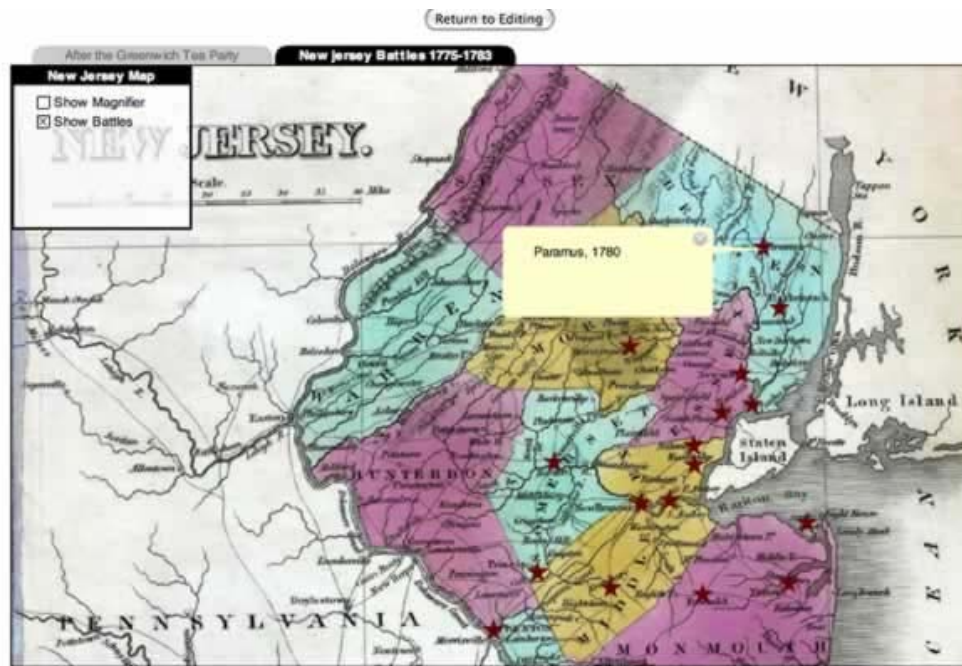
This tells VisualEyes to replace the placeholders (\$\$1,\$\$2,\$\$3) in INFOBOX:infoBattles with the values for the three variables (*lab*, *year*, *mark*, in that order) contained in the list \$battles.

- Save & Preview.
- Go to the New Jersey Battles tab and click on Show Battles.
- Click on the **dots** to make sure the infoBoxes are working. When clicked, each infoBox should display the location (*lab*), *year*, and *mark* for each battle (**dot**). If you look at the lower left corner, you'll see the value for \$\$click for each **dot**.



Now that you've checked your data, you don't need to display the mark values.

- Return to Editing
- Click on INFOBOX:infoBattles, and remove the third placeholder. The text should read: \$\$1,\$\$2
- Save & Preview, go to the New Jersey Battles tab, and click on Show Battles.
- Click on any **dot**. The infoBox will display the correct location and year for the battle.



Congratulations! You have created and edited an infoBox associated with **dots** on a **path**, and have increased your understanding of scripts in VisualEyes.

This completes the VisualEyes tutorial. You should be well on your way to using VisualEyes for your own projects.