NUON Application Tutorial

Contents

1. Pages and Widgets - Hello World.
2. Remote Control Events - Geography Quiz
3. Meet BOB - Geography Quiz Answers
4. Hunt The Wumpus - More Scripting

Lesson 1 – Hello World

Bowing to tradition, our first application simply puts the text string “Hello World” on the screen, and it stays there until the application quits. Here is how you make this NUON App.

Step 0 -- You need a PC running Windows 2000. Go to the vmlabs\experimental\UIFramework\Tutorial folder on your computer. It contains a batchfiles folder, and several lesson folders.

Step 1 – Navigate to the Lesson1 folder. Using your favorite word processor, open the lesson1.xml file – You should see this text.

```
<!-- Comment - My First NUON App -->

<snml>
    <page width="720" height="480" first-focus="tom">

        <text-box name= "tom" left="50" top="100" width="400" height="200">

                <text-line left="0" base="30">Hello World</text-line>

        </text-box>
    </page>
</snml>
```

Step 2 -- Close the file. Copy the bat files from the batch files folder to the Lesson1 folder. Open a DOS window, connect to the lesson1 folder, and type the batch command: *transcode lesson1*

Step 3 -- type the batch command: *compile lesson1*

This command uses the nmlc.exe tool to compile the lesson1.nml file into a binary format file called lesson1.npg The batch file creates a folder called compiled and puts lesson1.npg into the compiled/pages subfolder.

Step 4 -- Type the batch command: *bundle lesson1*

This batch command causes the ramload.exe tool to bundle your npg files and other resources such as images, scripts, and fonts into a file called ui.dat It also copies and renames the lesson1.npg file so that it is called startup.npg Finally, it combines the ui.dat file with a loader program to make a NUON.CD file.

Step 5 -- Type the batch command: *preview*.

This batch command causes the winpreview.exe tool to run. It will display two windows on your PC. One is labeled NUON Messages and the other is labeled NUON Screen. On the NUON Screen window, click on the FILE menu and choose Load NUON Page File. Use the load dialog box to choose the file lesson1/compiled/startup.npg

The NUON Screen window will display your Hello World message, verifying that the User Graphics part of your application works. You can quit the application by choosing Exit in the file menu.

Step 6 -- When you are satisfied by using the preview tool, that your UI is ok, you can go ahead and create a CD. Use any CD burning program to write the NUON.CD file that was created by the bundle batch program onto a CD.

Step 7 -- Load the CD into your NUON DVD player. It will load and display your Hello World message on the TV screen.

Explaining the steps:

Step 0. When a CD is loaded into a NUON DVD player, the player looks for a special file on the CD called NUON.CD. If it finds this file, it loads it into memory and executes it. The tools in the tools folder allow you to create this kind of CD with your application in the NUON.CD file. The NUON.CD file will also contains other code that your application is using to draw graphics, control the DVD player, and so on. This extra code is automatically included by the CreateNuiCD program.

Step 1. Each NUON application consists of a series of pages. Each page describes how the TV screen looks while that page is active. The page also describes the behavior of your application; what happens when a remote-control button is pressed, what happens when the DVD has finished playing a particular chapter, and so on.

Each page is a text file created with any text processor. The lesson1 page is already written for you. To create your own page, simply create a new folder and use a text processor to write your own page.

A NUON page is written in a language called NUON Markup Language (NML). This language conforms to the XML specification standardized by the World Wide Web Consortium. It is similar to HTML which is used to create Web Pages, but it is much easier to learn and to use in creating interactive applications. You can write directly in the NML language. However, you may find it even easier to write in another dialect of XML called Simple NUON Markup Language. The lessons in the tutorial use SNML.

line 1. Anything contained between the tags <!-- and --> is a comment. It is ignored by all the tools and never effects the behavior of your application.

line 2. Except for comments, your page must be included between the tags <snml> and </snml> The XML spec requires that every element on a page begin with a beginning tag and end with an ending tag. See the SNML Reference manual and NML Reference Manual for the set of allowed tags.

line 3. Every nml page has a <page> element that defines the part of the TV screen that is controlled when that page is active. This is done by providing a set of **attributes,** for example, width="720". The value of an attribute is always a string of text in quotes. The attributes are always contained in the beginning tag, that is they come after the opening < and before the closing >.

Each element has a set of allowed attributes. See the NML Reference manual (NMLR) for the names of the attributes allowed for each element.

In addition to the width and height attributes, the <page> element must specify the first-focus attribute which is the name of the **widget** that will first be in focus when the page is loaded and becomes active. When a widget is in focus, all **events** are first sent to that widget for handling before being sent to other widgets or other pages.

line 4. <text-box> is a kind of **widget.** See NMLR for a list of the kinds of widgets. All widgets can have name, left, top, width, and height attributes. The left and top attributes specify the position of the upper left corner of the widget on the page. The width and height attributes specify the size of the widget. The name attribute associates a name with the widget so that it can be referenced by other widgets. See NMLR for other attributes allowed for widgets.

line 5. A text-box widget can contain one or more <text-line> objects. A text-line describes a single line of text. The left and base attributes specify the position of the beginning of the line relative to the upper-left corner of the text-box that contains the line. The base attribute is the number of pixels from the top of the container to the baseline (the line defined by the bottoms of the capital letters) of the text string. The actual text is placed between the beginning and ending <text-line> tags. It is up to the author to make sure that your text string will fit on a single line in the position that you specified. Because no other attributes are specified the text is displayed as black letters on a line that is 24 pixels high. This is the **default** style of text. In Lesson 2, we will see how to use different text styles.

lines 6. Every element consists of a beginning tag, material contained in the element, and an ending tag. </text-box> is an ending tag, which is created by putting the / before the name of the element and enclosing it in <>.

Step 2. The batch files are provided to make it easy to use the tools in the tools folder. They should always be copied into the local folder where you are doing your work. The transcode batch file uses saxon.exe in the tools folder to convert your lesson1.xml file into the canonical form of xml called nml. The result will be a file called lesson1.nml in the lesson1 folder.

Step 3. The NML pages that you write to create your application are all simple text files, as are all XML documents. The NML compiler (nmlc.exe) is used to convert the text files to binary files that can be more efficiently loaded by the NUON DVD player. If you create an invalid NML page, the NML compiler will generate error messages.

When your application contains more than one nml page, you must use NMLC to compile each of them into an npg file.

Step 4. The bundle batch file collects all the files in the compiled folder and subfolders into a ramdisk image called ui.dat. It renames the file you designate on the command line (e.g. lesson1) to startup.npg Both the preview program and the NUON page loader expect that the first page will be called startup.npg. Other pages can be named anything you like. Finally, the ramdisk image is combined with other NUON libraries by the CreateNuiCD program and turned into a NUON.CD file that you can burn on a CD.

Step 5. The winpreview program is provided so that you can test and debug the user interface part of your application on a PC without actually making a CD and loading it onto a NUON DVD player. The NUON Screen that is displayed by the winpreview program has the same number of pixels as your TV screen. Colors of objects will not be exactly as they will appear on the NUON player, but they will be close. You can check out the layout of your pages and also navigation from page to page and focus change from widget to widget in response to events. In Lesson 2, you will see how the PC keyboard can be used to simulate DVD remote-control buttons. Of course, you can not actually play DVD movies using the winpreview application.

The winpreview NUON messages window displayed messages about errors that occur because of inconsistencies in your nml files. For example, if an attribute references a widget or page that is not defined anywhere in your app, an error message is generated.

Step 6. When you use the winpreview program, you may discover things you want to change in your NML pages. Each time you recompile, you should redo the bundle batch file which will create new versions of ui.dat and NUON.CD files. When you are satisfied with the winpreview results, go ahead and put the NUON.CD file on a CD. Using your own CD writer software on a personal computer, write NUON.CD to the top directory of a CD-R or CD-RW. The disc must use the ISO9660 disc format.

Step 7. The NUON.CD file contains your compiled application and VMLabs proprietary binary code used by your application to control the NUON DVD player.   It is designed to run on any NUON DVD player, but you may find that your application runs faster on some players than on others. If your application does some actions too fast, you will need to use timer widgets in your application to slow the actions down.

Lesson 2 - Geography Quiz

In this lesson, we introduce the text-menu widget which responds to arrow-key events by highlighting different items in a list. We also explore text-styles and colors.

Step 1 - Copy the bat files from the batch-files folder to the lesson2 folder. Open a DOS window, navigate to the lesson2 folder and execute the bat files: *transcode lesson2, compile lesson2, bundle lesson2*.

Step 2 - Execute the *preview* bat file and load lesson2/compiled/startup.npg. The NUON Screen window will show text boxes showing a quiz question and possible answers. The top answer is colored red instead of blue. Press the down-arrow key on your keyboard. This will cause the second answer to be highlighted instead of the first.

The text-menu widget responds to up and down arrow keys. Press the down arrow key until the fourth answer is highlighted. Now press the down arrow key again. This causes the lines to scroll so that answers two through five are shown rather than items one through four.

Step 3 - Burn the NUON.CD file created by bundle.bat onto a CD. Load the disk on your NUON player. You will see the same geography quiz page on your TV. Use the up and down arrow keys on your NUON DVD player to navigate up and down in the text box.

Explanation - Use a text editor to examine lesson2.xml. You will see this text (we have added line numbers in this document to make explanation easier).

```
1<!-- Lesson 2 A Geography Quiz -->
2<snml>
3 <page width="720" height="480" first-focus="answer">
4   <text-box  name="question"
5                         left="80"
6                         top="50"
7                         width="500"
8                         height="60"
9                         back-color="yellow">
10    <text-style name="Q" size="30" color="black" />
11    <text-line left="10" base="40" text-style="Q">The Capital of Uzbekistan is?</text-
line>
12   </text-box>

13   <text-menu name="answer" left="300" top="120" width="200" height="220"
  back-color="white" select-line="1" select-style="hilite">
14     <text-style name="normal" size="40" color="steelblue" />
15     <text-style name="hilite" size="40" color="#CD5C5C" />
```

```
16    <text-line left="0" base="50" text-style = "normal">Islamabad</text-line>
17     <text-line left="0" base="100" text-style = "normal">Ashkabad</text-line>
18     <text-line left="0" base="150" text-style = "normal">Tashkent</text-line>
19     <text-line left="0" base="200" text-style = "normal">Dushambe</text-line>
20     <text-line left="0" base="250" text-style = "normal">Kabul</text-line>
21   </text-menu>

22   <text-box name="next" left="80" top="360" width="500" height="70" back-
color="yellow">
23     <text-line left="10" base="40" text-style="Q">Next Question</text-line>
24   </text-box>
25  </page>
26</snml>
```

You can see that the general structure of this page is like lesson 1. Line 1 is a comment. Lines 2 and 26 are the beginning and ending snml tags.

Lines 3 and 25 show that this entire document describes a single snml page. The attributes on line 3 specify that the page dimensions are 720 pixels by 480 pixels and that the widget that will be the first focus of user-events is the widget named "answer".

This page contains 3 widgets. The first widget is a text-box described in lines 4 through 12. The opening tag for the first text-box is spread across lines 4 through 9. All of this information could also have been put on a single line. Breaking it up on multiple lines is done for readability. One place where you can not break up a line is when you are specifying text to be displayed. For example, if line 11 were broken between Capital and of, the line break would also appear in the text when it is displayed in your application. Or it might cause an error.

Line 9 illustrates a new attribute for a widget. The back-color attribute specifies a color to be used to paint the entire rectangle covered by the widget before adding any foreground information such as text. There are two ways to designate colors. The first way is to use a named color from the list included in the SNML reference. These names are recommended by the World Wide Web Consortium for use in XML vector graphics and in Cascading Style Sheets.

The second way of specifying a color is to actually provide values for the red, green, and blue components of the color. This must be done with a string of the form "#RRGGBB", where RR is a hexadecimal number between 00 and FF representing the red component, and GG and BB are hexadecimal numbers representing the green and blue components. For example, in line 15, the color indianred is represented by the string "#CD5C5C" instead of the name.

Line 10 shows the use of the text-style element. The text-style element allows you to specify the size, color, and other attributes of text. (See SNML reference for a complete list of text-style attributes). In line 10, we define a text-style with the name "Q" that specifies text that is to be 30 pixels high.

Line 11 describes a text-line to be displayed in the text-box. It specifies that the text-style to be used for display is the "Q" text-style. The text-line element also describes the position of the text-line in the text-box and the content of the line. As mentioned in lesson1, if you do not specify a text-style in a text-line, a default text-style is provided that specifies a line size of 24 pixels and black text. When you create a text-style, you only need to specify differences from the default text style described in the SNML reference manual. A text-style element can appear anywhere within a page description, it does not need to be local to a particular widget. For example, the "Q" text-style is also used in the bottom text-box widget in line 23.

In lines 13 through 21, we use the text-menu widget. This widget is like the text-box widget, in that you describe where it is on the page, and you describe where some text-lines are to be displayed in the widget and the styles to be used to display them. But the text-menu widget allows you to also specify a select-style and a select-line attribute, as shown in line 13. The text-line specified by the select-line attribute is displayed using the select-style rather than the style specified in the text-line element. In this case, select-line is set to "1". So the first text-line is displayed using the "hilite" style which uses the indianred color for text instead of the "normal" text-style which uses steelblue for a text color.

The text-menu widget also responds to up-arrow and down-arrow events. When you use the winpreview program, you can generate these events by pressing the up and down arrows on your PC keyboard. When you are running your application on a NUON DVD player, you generate these events by pressing the up and down arrows on the DVD remote control.

Note that the user events are sent to the widget on the page that has focus. In line 3, we specify that when this page is loaded, the first focus is on the widget named "answer". If we had specified the "question" on "next" widgets, events would be sent to those widgets and you would not see the highlight change in the "answer" text-menu.

By itself, the text-menu widget does not allow you to accomplish much. You can scroll up and down any number of lines (even more than can be displayed at once), but you can't use the highlighted line to do anything. This requires using a simple script language which is illustrated in lesson 3.

Lesson 3 - Meet bob

In this lesson, we use simple script commands to respond to user events.

Step 1 - Copy the bat files from the batch-files folder to the lesson3 folder. Open a DOS window, navigate to the lesson3 folder and execute the bat files. Note that this lesson has two xml files titled lesson3.xml and question2.xml. So we must execute *transcode lesson3, transcode question2, compile lesson3* and *compile question2*. This lesson also contains a script file called lesson3.bob. To process this file, we must execute the batch file *bob lesson3*. Then we execute *bundle lesson3*.

Step 2 - Execute the preview bat file and load lesson3/compiled/startup.npg We again see the geography question page. This time, instead of pressing the down-arrow, leave the first line highlighted and press the enter key. The result is a message saying that you are wrong. Now use the down arrow key to highlite the third line and press the enter key. You now get a message that you are correct. If you now press the enter key again, a second page is loaded with another question to be answered. This time, if you get the correct answer, you are asked if you want to exit the application. If you press the enter key, the application quits.

Step 3 - As in previous examples, you can burn the NUON.CD file created by bundle on a CD and load it into a NUON DVD player where it will show the same behavior as in the preview program.

Explanation - Use a text editor to examine lesson3.xml. You will see this text (again, we have added line numbers in this document to make explanation easier).

```
1<?xml version="1.0"?>
2<snml>
3       <script src='scripts/anml.bbo'/>
4       <script src='scripts/lesson3.bbo'/>
5       <page width="720" height="480" first-focus="answer">
6               <on-load>widgets.right.Hide(); widgets.right.HideBackground();
widgets.wrong.Hide(true);</on-load>

7   <text-box  name="question"
8                       left="80"
9                       top="50"
10                      width="500"
11                      height="60"
12                      back-color="yellow">
13    <text-style name="Q" size="30" color="black" />
14     <text-line left="10" base="40" text-style="Q">The Capital of Uzbekistan is?</text-
line>
15    </text-box>
```

```
16   <text-menu name="answer" left="300" top="120" width="200" height="220"
   back-color="white" select-line="1" select-style="hilite">
17       <on-click>chooseWidget(get_selected_line(widgets.answer), 3, widgets.right,
widgets.wrong)</on-click>
18     <text-style name="normal" size="40" color="steelblue" />
19     <text-style name="hilite" size="40" color="indianred" />
20     <text-line left="0" base="50" text-style = "normal">Islamabad</text-line>
21     <text-line left="0" base="100" text-style = "normal">Ashkabad</text-line>
22     <text-line left="0" base="150" text-style = "normal">Tashkent</text-line>
23     <text-line left="0" base="200" text-style = "normal">Dushambe</text-line>
24   </text-menu>

25   <text-box name='right' left='80' top="360" width="540" height="70" back-
color="yellow">
26       <text-line left="10" base="40"
text-style="Q">Correct! - Click for Next Question</text-line>
27       <on-click>frame.LoadPage("pages/question2.npg")</on-click>
28</text-box>

29   <text-box name="wrong" left="80" top="360" width="540" height="70" back-
color="red">
30     <text-line left="10" base="40" text-style="Q">Wrong! - Try Again</text-line>
31   </text-box>
32  </page>
33</snml>
```

Line 1 is an element enclosed in <?   ?>. This is an XML processing instruction. In this case it was generated by the XML text editor used to generate this file. Processing instructions are ignored by the NUON tools. While any text editor can be used to create XML files, there are several shareware and commercial editors that make it particularly convenient to create and maintain XML files. This file was created with a shareware program called HTML-Kit < http://www.chami.com/html-kit/>.

Line 6 shows the use of a new element in the page widget. The <on-load> element specifies a sequence of actions that are to be taken when the nml page is loaded. The actions are separated by semicolons and are enclosed between the opening <on-load> tag and the ending </on-load> tag. The actions are written in a scripting language called NUON Script. This is a simple object-oriented scripting language very similar to JavaScript. It was originally published by David Bets in Dr. Dobbs Journal (*reference?*) and was called BOB. Both names are used in this documentation.

The purpose of line 6 is to hide two widgets that are to be used for feedback. Normally, all the widgets on a page are displayed when it is loaded. But we do not want to immediately display the feedback text-box called "right". So we put the commands 'widgets.right.Hide()' and 'widgets.right.HideBackground()' in the <on-load> element. When a page is loaded, all the widgets on the page are placed in an array called widgets. You can reference any widget by using the form widgets.name where name is the name given to the widget in it's name attribute. So widgets.right.Hide() simply is a command to apply the built-in bob function Hide() to the widget named "right". (See NMLR for a list of the built-in bob functions that can be applied to widgets) A widget has both a background and a foreground that can be shown or hidden separately. In this case we want to hide the "right" foreground and background when the page is loaded.

We also want to hide the "wrong" feedback widget when the page is loaded. Here we show a shortcut for hiding both foreground and background by using the command widgets.wrong.Hide(true).

Line 17 is the next use of bob. In this line, we specify what action is to occur in the text-menu widget when the enter button is pushed on the remote-control. Again the action is described by a script contained between the opening <on-click> and closing </on-click> tags. The specified action is 'chooseWidget(get_selected_line(widgets.answer), 3, widgets.right, widgets.wrong)'. However chooseWidget is not a built-in bob function. It is a function written specifically written for this page and it is defined in a separate script file called lesson3.bob.

If you use a text editor to examine lesson3.bob you will see the following (again we have added line numbers in this doc).

```
1 define chooseWidget( choice, correct, trueWidget, falseWidget )
2 {
3       if( choice == correct )
4       {
5               falseWidget.Hide();
6               trueWidget.Show(true);
7               trueWidget.SetFocus();
8       }
9       else
10      {
11              trueWidget.Hide();
12              falseWidget.Show(true);
13      }
14 }
```

The purpose of this function is to provide feedback that a choice was correct or incorrect. If the choice parameter equals the correct parameter, the widget named by the trueWidget parameter is displayed and the falseWidget is hidden. Otherwise, the falseWidget is displayed and the trueWidget is hidden. In the case that the choice is correct, line 7 shows how the focus is also changed from the existing widget to the trueWidget. Otherwise, the focus is left the same, so that the user can try another choice.
Line 17 uses the built-in bob function get_selected_line to find out which line in the text-menu is currently selected. It passes that line number into the ChooseWidget function along with the number of the correct choice (3). It also passes in the name of the "right" text-box for the trueWidget and the name of the "wrong" text-box for the falseWidget.

The NUON runtime must know where to find the definition of the ChooseWidget function. Line 4 shows a <script> element that provides the path to find a script file that needs to be loaded whenever this page is loaded. Note that the path is to the compiled script file (extension .bbo) which was created by the bobc2 batch file.

Line 3 shows another script element referencing the file snml.bbo This file provides some functions that translate from the easy to author XML functions to the canonical bob functions. This file does not need to be included if you are writing directly in nml.

Line 27 shows another script. This specifies the action to be taken when a user has chosen the correct answer and the focus has now been transferred to the "right" widget. This widget displays the message to Click for the next question. When the user clicks, the <on-click> script specifies that a new page is to be loaded.

When a page is loaded, a special widget named frame is created that is used to contain the page. So the command 'frame.LoadPage("new page") ' is a built-in bob function that will replace the current page with a new page whose page is passed as an argument to the LoadPage function. If you examine the question2.xml file, you can see that the <on-click> element for the "right" widget specifies an action of 'frame.UnloadPage()' This simply unloads the existing page, but does not load another. The result is a blank screen.

Lesson 4 - Hunt The Wumpus

This lesson implements a game called Hunt The Wumpus. It shows the use of more scripting, and it is also written directly in NML, rather than the SNML dialect used in the previous lessons.

Step 1 - Copy the bat files from the batch-files folder to the lesson4 folder. Open a DOS window, navigate to the lesson3 folder. Note this lesson does not have any xml files, so you do not need to use the transcode tool. There are two nml files to be compiled: *compile startup* and *compile wumpus*. Use the bob batch file to compile wumpus.bob: *bob wumpus*. When you use the bundle tool, you must give it the name of the file to be used as startup, which in this case is named startup.nml. So you execute: *bundle startup*.

Step 2 - Execute the *preview* bat file and load lesson4/compiled/startup.npg Play a game of Hunt The Wumpus.
Rules:
Hunt The Wumpus challenges you to search through a maze of caves to find and shoot a Wumpus. At each cave, you are given a choice of 3 caves to go to next. Use the arrow keys to choose the cave you want to enter and press the Enter key. Some caves have pitfalls or bats that can seize you and end the game. You have 5 arrows to shoot into caves. If you think a Wumpus is in the same cave as you or in an adjoining cave, press the down arrow to enter the shooting frame and use the number keys to enter the number of the cave to shoot into. If you are successful, you will kill the Wumpus. If not, you will have used up one of your arrows. If you think the Wumpus is in a cave that does not adjoin the one you are in, you can shoot a smart arrow. Use the down arrow to enter the shooting frame and then use the number keys to enter the numbers of a path from your cave to the Wumpus cave.

Step 3 - As in previous examples, you can burn the NUON.CD file created by bundle on a CD and load it into a NUON DVD player which will allow you to play the game on your NUON DVD player.

Explanation startup.nml - Use a text editor to examine startup.nml Several new concepts are introduced, but the meaning of most of them is clear from the context.

Several elements serve the same purpose as elements in SNML, but have names more reflective of HTML. For example, <body>, and <group>. Also, instead of different widget elements, all widgets use the element tag <widget> with an attribute type that defines what kind of widget it is.

```
<widget type='text'
            textStyle='welcome'
            x='80'
            y='40'
            width='250'
```

```
                height='30'
                align='left'
     value='Welcome to "Hunt the Wumpus!"'/>
```

This is a new kind of text widget. It does not use text-line elements but specifies the text string in the value attribute. It also supports an align attribute.

```
<widget name='pic'
                type='image'
     src='images/wumpus.gif'
     x='80'
     y='120'/>
```

This is an example of an image type of widget. It allows you to specify a path to an image such as a gif file and instructions as to where to place it on the nml page. Note that lesson4 contains an images folder which contains the wumpus.gif image. The src attribute specifies a local path to navigate to that image file.

Explanation wumpus.nml - This page shows the use of more attributes in the text widget type.

```
widget name='exit1' type='text'
                x='10' y='10' width='40' height='28'
                textStyle='caveNumber'
                highlightTextStyle='hltCaveNumber'
                left='exit3'
                right='exit2'
                down='path1'
                onClick='game.Travel(0)'
                align='center'/>
```

This shows the use of the attributes to navigate amongst widgets.. left="exit3" means that pressing the left arrow causes Focus to leave this widget and to be transferred to the widget named exit3.

```
<widget name='path1' type='text'
                x='10' y='10' width='40' height='28'
                textStyle='caveNumber'
                highlightFrame='true'
                frameColor='yellow'
                bgColor='lightblue'
                maxLength='2'
                left='path5'
                right='path2'
                up='exit1'
                onClick='game.Shoot()'
```

```
                    align='center'/>
```

This shows that widgets of type "text" are also text-entry widgets. The maxLength attributes specifies the maximum number of characters that can be entered.

Explanation wumpus.bob - This script file implements the logic of the game. It shows the use of the object oriented bob language to create objects like Caves and Bats. It also shows how to define methods for these objects.

It also defines functions that govern what happens when a player shoots an arrow or moves from cave to cave.