exsys Corvid How To

Expertise at Your Fingertips www.exsys.com

Designing Systems to Run in a Kiosk Environment

A system demonstrating this "How To" can be run from the "Kiosk" section of:

http://www.exsys.com/support/howto

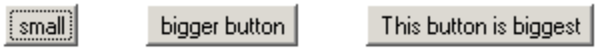
The code for the sample system can be downloaded from the same page.

A kiosk is an electronic service booth that customers can interact with, similar to an ATM or e-ticket check-in.

A kiosk that will have a touch screen monitor, with no keyboard or mouse accessible to the user, needs to do something special to make it easier for users to answer questions by tapping their finger on portions of the screen. This works best by using big buttons for the answers or big hot spots in an image map. The 'kiosk' demo shows how to do both when running as an applet or application. (A kiosk that will have a keyboard and mouse available to the user does not need to do anything special.)

Having touch screen capability does not require special software as far as Corvid is concerned. This capability is provided by the operating system which makes the user touching the screen appear to Corvid as a mouse click.

To use buttons to ask the questions when running as an Applet, under the 'Ask With' tab in the 'Variables' window, select 'Button' for each question that will be asked of the user. The buttons will be automatically resized to fit the largest text of the question's values. To make the buttons wider, add more text. Exsys Corvid will strip leading and trailing spaces, but adding spaces preceded and followed by a non-space character will make the button larger.



This will use buttons, but they may not be as large as desired.

To have the applet display a larger button, use image maps. Create an image in a graphics design program such as Photoshop, to use to ask the question. This should have regions for the user to press or click to answer the question. These do not actually have to be designed as "buttons" and can just be a region of the screen that indicates a particular answer.

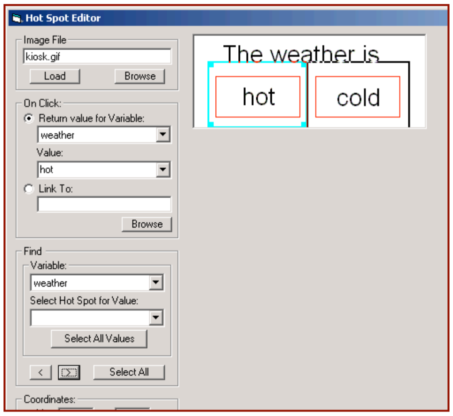

Hot Spot Editor

Image File: kiosk.gif

On Click:
● Return value for Variable: weather
Value: hot
○ Link To:

Find
Variable: weather
Select Hot Spot for Value:

Coordinates:

In the Variable window, select the question associated with the image and click the “Ask With” tab. Click “Image Map” and then click “Edit map”. Select the image file created for the question, and add the “hot spots”. (See the “Image Maps” section of chapter 5 of the Exsys Corvid manual for details).

To have big buttons when running as a Servlet (Corvid Servlet Runtime), normal HTML can be used to create buttons the size you want or you can use HTML Image Maps.

Run the demo system to see an example of big buttons on an image map when running as an applet. Open the system with Corvid and under the 'Ask With' tab click 'Image Map'. Click on the '>' button to select the hot spots. You can make the hot spots as big as you wish.

Typing into An Edit Box

Since a kiosk touch screen typically does not have a keyboard, the user cannot type text into edit boxes. The 'kiosk' sample KB shows how to overcome this limitation when running as an applet. Run it and see how you can type the number using your mouse (or your finger on a touch screen monitor). Each time you click on a digit, that digit is added to a String representation of the number. The “Back” button erases the last character entered. When you click on 'Enter', it converts the String to a Numeric.

How This is Done

Open 'kiosk.CVD' in Corvid and edit the [_numeric_image_map] variable.

Under the 'Ask With' tab, click 'Image Map'

Click on a few hot spots covering the digits. Notice the “return string” is the digit represented by the button. To build the full string, that digit must be concatenated to [_numeric_text], which holds the string representation of the number typed in so far.

This is repeated until the user clicks on the 'Enter' button. Notice there are two hot spots for 'Enter'. One returns the empty string for [_numeric_image_map] and one returns "Yes" for [_numeric_done].

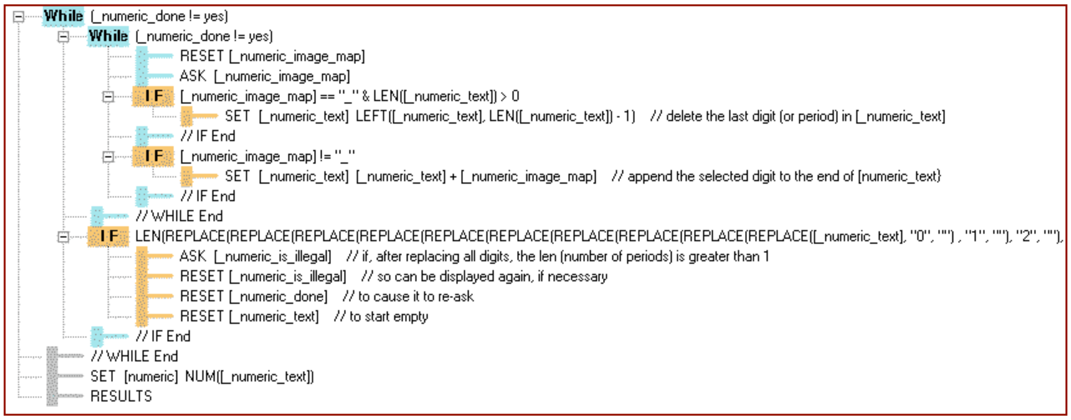
Each digit has to be combined with the previously selected digits. This is performed in the Command Block named 'Ask A Numeric'.

The user can erase the last digit (or period) that was entered by clicking on the Back button in the Image Map. Instead of returning a digit, the Back button returns a "_". When the command block detects a "_", it deletes a character instead of appending one.

It stops when [_numeric_done] has a value. The Command Block's logic is:

While not done, get the next digit and concatenate it to the end of [_numeric_text]. If user typed a bad numeric (such as 3.4.5 which has too many decimal points), then report error, reset and start over.

The Command Block has comments that explain the purpose of each command.



```
While (_numeric_done != yes)
    While (_numeric_done != yes)
        RESET [_numeric_image_map]
        ASK  [_numeric_image_map]
        IF  [_numeric_image_map] == "_" & LEN([_numeric_text]) > 0
            SET  [_numeric_text]  LEFT([_numeric_text], LEN([_numeric_text]) - 1)    // delete the last digit (or period) in [_numeric_text]
        // IF End
        IF  [_numeric_image_map] != "_"
            SET  [_numeric_text]  [_numeric_text] + [_numeric_image_map]    // append the selected digit to the end of [numeric_text}
        // IF End
    // WHILE End
    IF  LEN(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE([_numeric_text], "0", "") , "1", ""), "2", ""),
        ASK  [_numeric_is_illegal]    // if, after replacing all digits, the len (number of periods) is greater than 1
        RESET [_numeric_is_illegal]    // so can be displayed again, if necessary
        RESET [_numeric_done]    // to cause it to re-ask
        RESET [_numeric_text]    // to start empty
    // IF End
// WHILE End
SET  [numeric]  NUM([_numeric_text])
RESULTS
```

The inner While loop gets each digit and appends it to [_numeric_text] until the user clicks on the 'Enter', which sets [_numeric_done] to “yes”. If the user typed a valid number, then the outer While loop terminates and the string is converted to a numeric. But if the number is invalid, it resets all the variables involved (which initializes [_numeric_done] back to “no”) and the outer loop repeats everything again. The number is invalid if the number of periods is more than one, if the period is the only character, or if the string is empty (the user hit Enter without typing any digits or periods).

For example, these are not valid numerics:

123.45.6	Too many periods
.	No digits
…….	Too many periods and no digits
.098.765.54	Too many periods

The IF condition determines if the numeric is bad. After replacing all digits with an empty string (which deletes the digits), the length of the remaining string is the number of periods. The IF condition tests if the count of periods > 1 or no digits. If the numeric is invalid, [_numeric_is_illegal] is asked. Since that Static List variable has only one value, “OK” and asked using Buttons, it appears as a message. When the user hits “OK”, the variables involved are RESET.

After the user enters a valid number, the text they “typed” is converted to a number by using the NUM() function and assigned to [numeric].

This approach will work when running the system with the Corvid Applet Runtime or as a standalone application. This approach will not work as well using the Corvid Servlet Runtime since the screen would have to be redrawn after each button click. For the servlet, Javascript could be used to validate the input on a single screen, with the final value passed to Corvid.

Some operating systems such as “Windows Tablet” have virtual keyboard functionality built in and would not require any special additions to the Corvid system.