Creating an Image Map

By: Mark Lee

January 14, 2008
Download a zip file containing an example

An image map is a picture with clickable regions. Image maps are typically found on WebPages as navigational tools, allowing users to click on pictures or sections of a picture to go to other pages. But, an image map can be used for a lot of other things.

A Few Examples

An image map could be used to simulate irregularly shaped buttons.

An image map could be used in a point and click type game, such as the popular Nancy Drew series. The player would navigate rooms, pick up objects, dial numbers on a cell phone, etc ... just by clicking the right region of the pictures.

Any map could be used to learn geography. The program would ask the user to locate something on the map, and the user would simply click in the right region.

Young kids could learn their body parts by clicking on an image of the human body. This demo shows such an application.

How To Create an Image Map

Create an image with solid colored regions. Bitmaps (.bmp) take up a lot of space, so using a compressed format may be the better option. Either .gif or .png would be an appropriate choice. Since native Liberty BASIC only supports bitmaps, a 3rd party dll is required for loading another image format. I used PBImage DLL by Dean Hodgson to load a png file.

The next step is to convert the RGB color codes of the solid color regions to single numbers, which are used by windows to read pixel values. I used the following code to do this.

```
[begin]
input red
input green
input blue
print RGB(red, green, blue)
input again$
if again$ = "y" then [begin]
end
Function RGB(red, green, blue)
 RGB= red + (green * 256) + (blue * 65536)
End Function
```

Now, write a program to scan each pixel in your image. The scanned color is then compared to the previously selected colors in a Select Case routine, allowing the program to assign body parts to each pixel. Colors that are not part of the clickable regions are grouped together in a Case Else statement. Store this information in a Random Access File database. The following code snippet differs slightly from the code in the download, but it demonstrates the scan process.

```
'index the pixels
PixelNumber = 1
for yVar = 0 to 629
'Cycle through the height of the image to be scanned
for xVar = 0 to 319
'Cycle through the width of the image to be scanned

 pixelLong = pixelLong(xVar, yVar)

Select Case pixelLong

 Case 8421504 ' Face

Part$ = "1"
 PUT #Body, PixelNumber
 PixelNumber = PixelNumber + 1

 Case 16711553 ' Neck

 Part$ = "2"
 PUT #Body, PixelNumber
 PixelNumber = PixelNumber + 1

 Case 16512 ' Shoulders

 Part$ = "3"
```

```
PUT #Body, PixelNumber
PixelNumber = PixelNumber + 1

Case 16646144 'Chest


Part$ = "4"
PUT #Body, PixelNumber
PixelNumber = PixelNumber + 1

Case 254 ' Arms

Part$ = "5"
PUT #Body, PixelNumber
PixelNumber = PixelNumber + 1

Case 131071 ' Hands

Part$ = "6"
PUT #Body, PixelNumber
PixelNumber = PixelNumber + 1

Case 16711807 ' Stomach

Part$ = "7"
PUT #Body, PixelNumber
PixelNumber = PixelNumber + 1

Case 16744448 ' Hips

Part$ = "8"
PUT #Body, PixelNumber
PixelNumber = PixelNumber + 1

Case 98304 ' Legs

Part$ = "9"
PUT #Body, PixelNumber
PixelNumber = PixelNumber + 1

Case 8388863 ' Feet

Part$ = "10"
PUT #Body, PixelNumber
PixelNumber = PixelNumber + 1
```

```
 Case Else 'Non Body Part

 Part$ = "0"
 PUT #Body, PixelNumber
 PixelNumber = PixelNumber + 1

 End Select

next xVar
next yVar
```

Such a program shouldn't take too long to index even large pictures. After certain modifications were made to my program to speed up the process, it took my computer less than a minute to scan a little more than 200,000 pixels. (That's a 320 x 630 size image.)

The use of a few API/DLL calls, including the GetPixel function, will be required. It is also necessary to retrieve the device context of the graphic box. Release that device context before exiting the program to free memory.

```
Function pixelLong(hDC, xVar, yVar)

    CallDLL #gdi32, "GetPixel",_
        hDC as Ulong,_
        xVar as Long,_
        yVar as Long,_
        pixelLong as Long
End Function

Function hDC(hGBox)

        CallDLL #user32, "GetDC",_
        hGBox as Ulong,_
        hDC as Ulong

End Function



Function LoadImage(index, file$)

calldll #pb, "IxGetImage",_
index as long,_ 'an index number assigned by the programmer
file$ as ptr,_  'name of image file on disk
```

```
hImage as ulong 'returns handle of image in memory

loadbmp "pic", hImage

End Function



' Now release the DC just once at the end of the scan
    CallDLL#user32,"ReleaseDC", _
        hGBox as Ulong, _
        hDC as Ulong, _
        result as Long

    wait

    [quit]
    Unloadbmp "pic"
    Close #Body
    Close #t
    Close #pb
    End
```

The following code snippet shows the Image Map program obtaining the body part information from the Random Access File created by the Image Scan Program. The clicked color is compared to the stored colors using a Select Case routine. Colors (or regions) that aren't part of the clickable regions may either be grouped together in a Case Else statement or ignored completely. If you want to address an incorrect click or show a message saying that the user clicked on a non clickable region, use the Case Else option.

```
[go]
    xCur = MouseX
    yCur = MouseY

    if yCur > 0 then RecordNum = (yCur * 317) + xCur + 1
    if yCur = 0 then RecordNum = xCur + 1


    Gettrim #dat, RecordNum


    Select Case Part$

        Case "0"
```

```
        #t.box, "Drawbmp face 0 0; Discard"

        Case "1"
        #t.box, "Drawbmp neck 0 0; Discard"


        Case "2"

        #t.box, "Drawbmp shoulders 0 0; Discard"
        Case "3"


        #t.box, "Drawbmp chest 0 0; Discard"
        Case "4"

        #t.box, "Drawbmp arms 0 0; Discard"
        Case "5"


        #t.box, "Drawbmp hands 0 0; Discard"
        Case "6"


        #t.box, "Drawbmp stomach 0 0; Discard"
        Case "7"


        #t.box, "Drawbmp hips 0 0; Discard"
        Case "8"


        #t.box, "Drawbmp legs 0 0; Discard"
        Case "9"

        #t.box, "Drawbmp feet 0 0; Discard"

        Case "n" ' No Body Part

    End Select


Wait
```

Each clickable region can have a separate picture associated with it. The new picture may replace the old picture when the mouse moves over the region or when a mouse button is either pressed or released. For that matter, a different picture could be triggered with each of the mentioned events. Options at this point

are only hindered by your imagination.

What else could you do with this demo? It could be in the form of a game where the user has to click on a particular body part to get the question right, or it could simply be used in Flash Card mode, where the program tells the user what body part he/she just clicked on. Either way, the click can trigger more than just a graphic change. Text and sounds may be triggered. In fact, anything Liberty Basic is capable of doing can be triggered with an image map click.

The Image Scan program included in the download at the top of this article demonstrates the scan step, and the Image Map program demonstrates using the stored information to simulate an image map.

I would like to thank Alyce Watson for the API and GDI knowledge she so generously shares, Dean Hodgson for the DLL used in my code example, which can be downloaded from |Alyce's site, and I would like to thank Janet for her suggestions and tips.