PHILIPS

PHILIPS INTERACTIVE MEDIA

Technical Note #82

A Graphical Method for Hotspot Generation

Graham Trott, BEPL

October 10, 1992

This note describes a graphical method for defining hotspots and an MPW script for generating resource compiler source or any other source that acts as input to the CD-I build. This method uses the **depict** MPW tool, part of Palomar Software's PICT Detective.™

A GRAPHICAL METHOD FOR HOTSPOT GENERATION

Hotspot definitions are, let's admit it, a pain—with all those fiddly numbers that, if mistyped, are virtually impossible to find. Here's a way of defining them graphically and then automating the production of the program script, whether it be for Ken Ellinwood's Resource Compiler (rc) or for some other custom method.

The easiest way of defining hotspot locations is by drawing them over the image to which they belong. If you use a layered Macintosh drawing program (I use Deneba's UltraPaint since it's cheap and works in pixels), you can draw a set of open frame rectangles over your image. You can then ask the program for the coordinates and sizes of each rectangle and transfer them to your CD-I script.

I got fed up with all this manual effort and was about to embark on some custom Mac software when APDA announced PICT Detective, a package that analyzes PICT files and delivers a report. It's a simple matter to create an MPW shell script that takes the output and extracts the information about the rectangles. All other object types (primarily the image itself) are ignored.

The script that follows embeds a call to depict (the MPW tool from PICT Detective) and then extracts hotspot information from the result. With a little effort, you can adapt the script to generate rc source or anything else that acts as input to the CD-I build. The object is to avoid transcribing numbers; this improves reliability and makes changes far less painful.

If you have comments or questions about the information in this note, contact Graham Trott at BEPL (e-mail address: gtrott@applelink.apple.com).

Sample MPW Shell Script

```
if (#) == 0
   echo "Hotspot generator, V1.00 from GT Computing"
   echo
   echo "This script examines a PICT file and extracts FrameRect"
   echo "objects, which it assumes to be hotspot rectangles."
   echo "These are output as source lines for the TVO object list"
   echo "compiler (also from GT Computing)."
   echo "To create a hotspot list, first convert your image into a"
   echo "PICT file, then load it into a suitable drawing package"
   echo "as a background (PhotoShop is no good here, but any layered"
   echo "drawing package such as UltraPaint will do).  Now draw the"
   echo "hotspot rectangles where you want them to be on the image"
   echo "and save the file.  Run 'hotspots' to generate the list."
   echo
   echo "Syntax:"
   echo "hotspots [options] <pictfilename> [options]"
   echo
   echo "Options:"
   echo "If no options are given, the output will be a line of data"
```

```
   echo "for each rectangle, in the following form:"
   echo
   echo "selection HOTSPOTn at <left> <top> from rectangle size <width>
<height>"
   echo
   echo "-n <name>       replace HOTSPOT with <name>"
   echo "-X <data>       replace <left> with <data>"
   echo "-Y <data>       replace <top> with <data>"
   echo '-x <data>       prefix <left> with "<data>+"'
   echo '-y <data>       prefix <top> with "<data>+"'
   echo
   echo "The syntax of this script and the form of the data it outputs"
   echo "are peculiar to the needs of the author's hotspot management"
   echo "system and will probably need to be adapted."
   echo
   exit
end
set HSName "HOTSPOT"
set PositionX ""
set PositionY ""
set OffsetX ""
set OffsetY ""
loop
   break if "{1}"=="
   if "{1}"=="-n"
      shift 1
      set HSName {1}
   else if "{1}"=="-X"
      shift 1
      set PositionX "{1}"
   else if "{1}"=="-Y"
      shift 1
      set PositionY "{1}"
   else if "{1}"=="-x"
      shift 1
      set OffsetX "{1}+"
   else if "{1}"=="-y"
      shift 1
      set OffsetY "{1}+"
   else
      set FileName {1}
   end
   shift 1
end
set newwindowrect 100,100,100,100
open temp -n -t
depict {FileName} -d -hide bitmap -hide clut -hide PicComment >temp
find •:"/FrameRect ∂(∂(/"
if {status}!=0
   echo "No rectangles found!"
   close -n temp
   exit
end
replace § ""
loop
   find "/∂)∂)/":"/FrameRect ∂(∂(/"
   if {status}!=0
      break
```

```
   end
   replace $ ∂n
end
find "/∂)∂)/":∞
replace $ ∂n
find •
replace / / "" -c ∞
set index ∅
for item in `catenate temp`
   (evaluate "(item)"=-/(=)®1,(=)®2,(=)®3,(=)®4/) >Dev:Null
   evaluate top=(®1)*2
   evaluate left=(®2)*2
   evaluate height=((®3)-(®1))*2
   evaluate width=((®4)-(®2))*2
   if (left)<∅
      set left ∅
   end
   if (top)<∅
      set top ∅
   end
   if "(PositionX)"!=""
      set left "(PositionX)"
   else if "(OffsetX)"!=""
      set left "(OffsetX)"(left)
   end
   if "(PositionY)"!=""
      set top "(PositionY)"
   else if "(OffsetY)"!=""
      set top "(OffsetY)"(top)
   end

# The following line generates the output from the utility
# and should be amended to whatever output format you require.

   echo selection (HSName)(index) at "(left)" "(top)" from rectangle size
(width) (height)
   evaluate index+=1
end
close temp -n
```