Netgem i-Player TV-Friendly Content Design Guidelines

Author: SCR

Document printed on: Tuesday, August 10, 2004

Revision: 113

Table of Contents

I INTRODUCTION

Interactive TV services extend the traditional one-way mode of watching and using the television to the realm of user-driven applications commonly associated with the Internet. There are very few standards for producing Interactive TV applications and for making them available to set-top boxes and most of them require complex authoring applications and expert developers. In contrast, the Internet has well-established standards for authoring and publishing applications, including multimedia content. In addition, the Internet has by far become the preferred media for making content and applications available to the public.

That is why Netgem I-Player is an open platform capable of accessing, loading and displaying standard web pages via standard Internet protocols and formats. Authoring for I-Player doesn't require much more than authoring for personal computers. In fact, most web pages can be accessed and viewed "as-is" on the television through I-Player as long as they don't rely of specific PC components and extensions. Therefore, there are two types of I-Player Interactive services:

- Those originally designed for the PC and adapted for the television, and
- Those created specifically for the television

Clearly the first step for application developers, editors and content providers to extend their reach TV viewers is to adapt existing content in a manner that makes it a positive experience. For that, it has to be readable, usable and, when appropriate, entertaining. This can be easily achieved by integrating some facts (such as the dimensions of the TV display) and guidelines.

The second step is to fully rethink the ergonomics and interface of the service or application in the context of the television, taking into consideration the differences between the attitude of TV viewers and PC Internet users. In addition, services designed specifically for Interactive TV can integrate the actual TV picture or can appear like OSD menus, overlaid over the current TV program.

Finally, some general optimization factors allow Interactive TV services to be more enjoyable and more usable. They include the display quality, the choice of colours, the ease of navigation, the load and rendering speed and, when appropriate, the integration of the TV program.

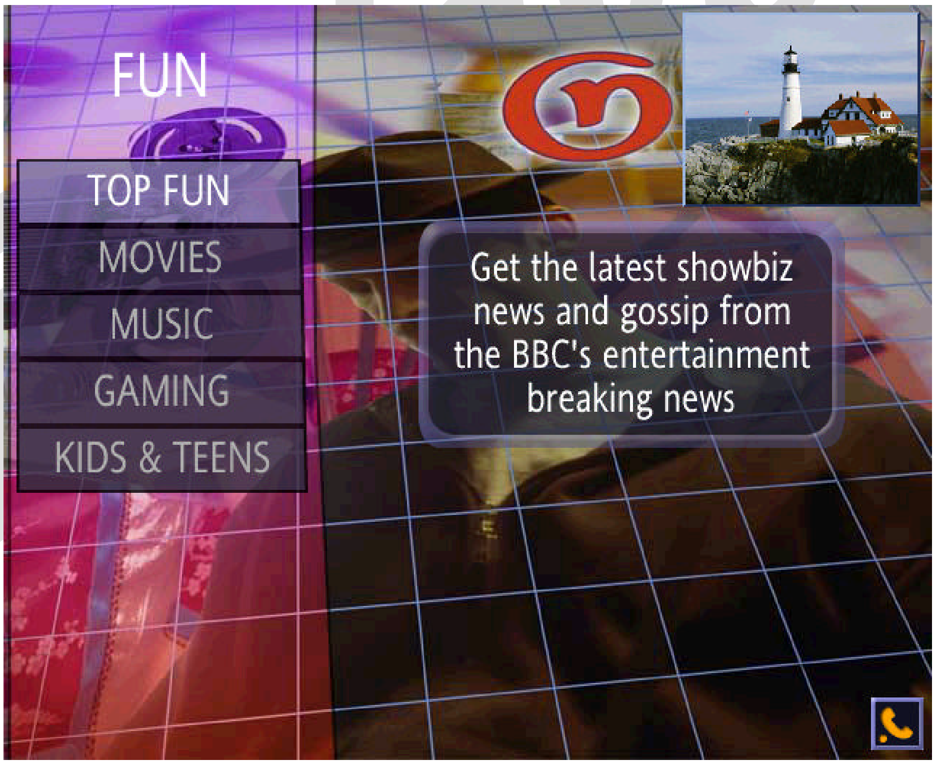

FUN
TOP FUN
MOVIES
MUSIC
GAMING
KIDS & TEENS
Get the latest showbiz news and gossip from the BBC's entertainment breaking news

Figure 1: Example of Interactive Menu

II DISPLAY CHARACTERISTICS

II-1 *Resolution*

TV set characteristics limit the capabilities of any device displaying on the TV. Because Netgem I-Player platforms (hardware and software) have been designed and optimized to display on the TV set, they achieve excellent quality in spite of many inherent TV limitations.

Unlike PC monitors that have a higher and adjustable refresh rate, TV sets have a fixed resolution, which depends on the local TV standard. Taking into account the safety margins around the image, Netgem I-Player platforms achieve a resolution of 640 x 520 pixels in PAL mode (Europe) and 640 x 480 pixels in NTSC mode (North America).

Like most browsers, I-Player automatically adds margins to an HTML page. The default top and bottom margins are 10 pixels high and the default left and right margins are 6 pixels wide. These default dimensions can be overwritten using the HTML <BODY> attributes TOPMARGIN, BOTTOMMARGIN, LEFTMARGIN and RIGHTMARGIN. However, it is not recommended to set them to 0 as it would make the page content (text or images) cling to the borders of the display.

In fact, very few assumptions need to be made about the screen resolution. There are some standard HTML techniques for optimizing the display of web pages without having to write hard-coded pixel dimensions in table rows and columns. These techniques are described later.

III WRITING INTERACTIVE TV CONTENT FOR I-PLAYER

III-1 *Introduction*

The Netgem i-Player browser was designed to render standard HTML pages as close as possible to their original design. However, a few considerations must be taken into account when designing pages for i-Player.

The most important aspect to consider is that each browser has its own rendering engine regardless of the supported standards, norms and compatibilities. While the actual standards describe functionality, they don't enforce a specific implementation. This is the reason why not all PC browsers are compatible. In fact, some browsers like Netscape and Microsoft Internet Explorer set many norms, which sometimes became official specifications, simply because they had chosen to implement such and such aspect of a standard.

The i-Player browser rendering engine performs the task of interpreting the content pages (HTML, CSS and JavaScript) and of displaying them on the TV. This task is sometimes called repurposing, though repurposing refers to actually converting for use in another format. While the i-Player browser is compatible with the latest open standards (HTML, CSS, DOM and JavaScript), its rendering of standard web pages may not be the same as the one obtained by other web browsers (PC or TV browsers). It doesn't mean that it doesn't correctly support these standards. It means that given the constraints of the low-resolution display (compared to PC monitors), its heuristics for interpreting the official web standards are fine-tuned to obtain a readable page where the user can navigate.

We therefore encourage web developers who want to design Interactive TV content for i-Player to start with the assumption that it supports open Web standard and to take into account the fact that the TV screen is not very large (in terms of pixels).

III-2 *General considerations*

- Different browsers editors may have different rendering of the same page
- The size of the fonts used by I-Player to print text on the television is by default larger than the one used by PC browsers
- The user can change the overall size of the text printed by i-Player (to improve readability on small TV sets)
- The user cannot change the size of the display screen (the window of PC browsers can be resized manually)
- Different television may have different colour settings and therefore images may not look the same (in particular high contrast and colour saturation)
- When a page is too wide and i-Player is unable to fit it within the width of the TV screen, automatic horizontal scrolling is enabled
- Long pages can be viewed by scrolling vertically
- The user navigates on the page from link to link, moving the selection up, down, left or right
- I-Player doesn't have the CPU power of a modern multimedia computer so pages with heavy graphics may slowdown rendering

IV OPTIMIZATION FACTORS

IV-1 *Minimize page download and rendering time*

Keep pages as short as possible

The shorter they are, the faster they are downloaded and displayed. What matters is the actual HTML file length so even extra empty lines or extraneous spaces (for instance between tags) as well as HTML comments should be removed. For instance, a simple cleanup of a standard cnn.com page can reduce its size by more than 10%.

Keep the size of image files as small as possible

Each image file format (GIF or JPEG) works best with certain types of images. In general, photographs should use JPEG format (with some compression) while drawings, logos and glyphs should use GIF format. In any case, use the HEIGHT and WIDTH attributes with the <IMG> tag to speed up the rendering. Most image manipulation tools can automatically optimize the quality to size ratio.

Reuse images

By taking advantage of the standard caching capabilities, page download and rendering time can be significantly shortened when images are reused. This means using the same URLs to reference images in different pages (for example a logo or a navigation bar). If parts of an image change according to the current page (highlight of the current item in a navigation bar), it is preferable to either splice the image or to use the non-highlighted image in a table background and superimpose the highlighted part.

IV-2 *Ensure that the display quality is optimal*

Always test preliminary work directly on the television

Because rendering on the TV screen is significantly different from the PC browsers, the only way to ensure that the result meets the expectations is to use a Netgem I-Player platform and a medium size TV set. It is also important to check the result (especially readability) from the normal TV viewing distance (about 3 meters, or 10 feet).

Use large fonts

I-Player uses a default font size that is larger than the one used by PC browsers. By forcing the font size to a smaller one, some text may become difficult to read.

Do not use saturated solid colours

They usually create artifacts such as bleeding and flickering. For instance, rather that using a pure white (#FFFFFF), use a light gray (#EFEFEF).

Avoid high contrast horizontal separations or lines

They can cause unpleasant flickering.

IV-3 *Simplify content presentation*

Layout content using tables rather than frames

Frames usually take longer to load and to render. In addition, they make navigation more difficult, especially for novice users. Frames do present advantages in some cases, but when appropriate, it is recommended to convert frames to tables.

Size table and cell widths proportionally

It is best to use percentages when partitioning table columns rather than fixed pixel width. It gives the browser more flexibility in optimizing the layout in case the content of some cells exceeds the available dimensions. Use the WIDTH=100% attribute in the <TABLE> tag to enlarge the table to the entire screen width (the last column clings to the right margin.). If you prefer to add some extra margins, use WIDTH=96% for instance.

Avoid graphics that contain mostly text

Instead, use actual text (you can change the colours, the font size and the style). The problem with text in images is that it is often too small and very difficult to read.

Align and uniformly size controls

Use the WIDTH=100% attribute in the <INPUT> tag (form fields) to enlarge controls (text input, text areas, select lists and buttons) to the entire available horizontal space in a table cell). It produces a more consistent and pleasant layout.

IV-4 *Simplify navigation and form handling*

Always use client-side image maps

Server-side maps (ISMAP) require the user to select a spot (a specific pixel in the image) by moving the arrow cursor. Rather, use client-side image maps (USEMAP) where each area can be easily selected.

Keep HTML forms as simple as possible

Complex forms require puzzling decisions. Try to make forms fit on one screen without scrolling vertically so that the submit button is always visible. Split longer forms into two or more short pages.

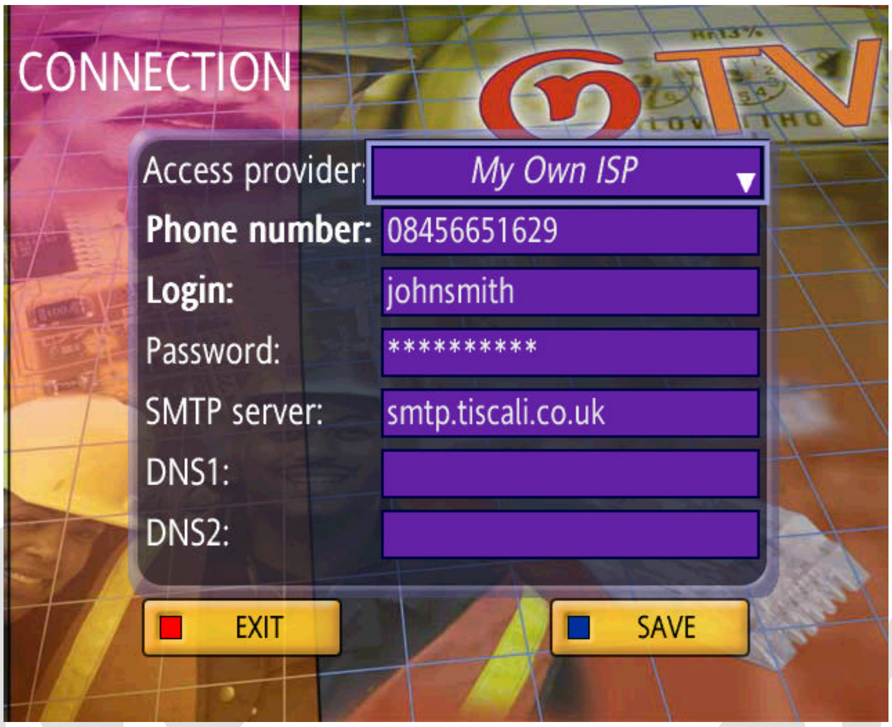
Simplify form submission when using image buttons

For image input <INPUT TYPE=IMAGE>, you use the "NOCURSOR" attribute to prevent the cursor from being displayed. It automatically validates the input (submits the form), as if the center of the picture had been selected. In most cases, the actual coordinates of the selected pixel are irrelevant to the form.

Also, because small TYPE=IMAGE buttons are almost always used as clickable buttons (regardless of which pixel of the image is actually clicked), image buttons that are less than 97 pixels wide and less than 65 pixels high are automatically submitted when the user presses the OK button.

IV-5 *Use standard and extended CSS attributes to enhance the look of the page*

As an example, the page below (Figure 2) is a simple form, using customized input fields (SELECT, TEXT, PASSWORD and BUTTON).



CONNECTION

Access provider: My Own ISP

Phone number: 08456651629

Login: johnsmith

Password: **********

SMTP server: smtp.tiscali.co.uk

DNS1:

DNS2:

Figure 2: Simple form using customized styles

Use non-tiled background in pages and tables

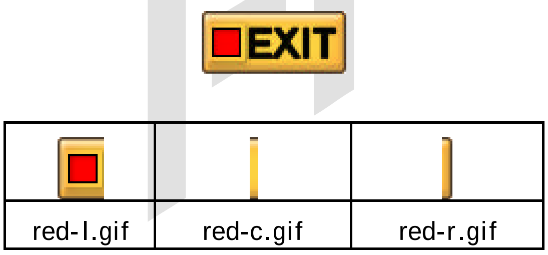
There are many instances where the page contains a banner that defines the background of the top part of the page. It is important to be able to use an image in the page background (like a watermark) without having the image tiled vertically (or horizontally).

To put an image into the page background and preventing it from being tiled (replicated), use the "background-repeat:no-repeat" style of the BODY or TABLE.

Change the default gray aspect of controls (<INPUT>, <SELECT> and <TEXTAREA>)

By redefining the styles of INPUT elements, you can achieve

Use resizable, bitmap buttons



```
.redButton {
  height: 43;
  color: black;
  font-size: 15pt;
  text-align: center;
  text-shadow: none;
  border: none;
  padding: 0 20 0 45;
  vertical-align: middle;
  background-color: transparent;
  background-image: url(red-c.gif);
  background-repeat: three-cut;
}
```

Figure 3: Bitmapped, auto-sized buttons and the 3 bitmap elements

In addition to changing the system colours to obtain standard buttons with different hues, I-Player extended CSS styles can also instruct the display manager to use special auto-sized bitmapped buttons. They are a combination of bitmapped buttons (a background image is used for the shape) and standard buttons because the text (caption) is not part of the image and because the size (width) of the button is automatically adjusted to fit the caption.

To each bitmap button correspond three bitmaps: the left edge, the right edge and the body (middle part). These bitmaps must be named BMPNAME-x.gif, where x must be 'L' for the left edge, 'C' for the center and 'R' for the right edge. The height of the button must be explicitly set in the class.

The extended style attribute is "background-repeat: three-cut".

Use bitmapped checkboxes and radio buttons

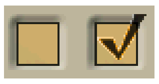
	`INPUT[TYPE=CHECKBOX] { checked-image:url(checkyellow-on.gif); unchecked-image:url(checkyellow-off.gif); }`

Figure 4: Bitmapped check box (checked and unchecked)

For checkboxes, more refined look can be achieved when using a bitmap for each of the checked and unchecked states. The page can use the checked-image and unchecked-image extended CSS attributes for the <INPUT TYPE=CHECK> or <INPUT TYPE=RADIO> tags to specify the bitmaps.

Use special "box" background in tables and table cells

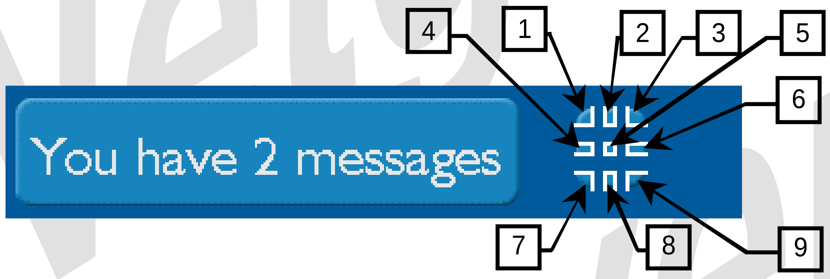


Figure 5: Table with background-repeat: nine-cut and the nine tiles

Like the page body background image, the HTML semantics of the table or table cell background imply that the background image be tiled horizontally and vertically to fill the entire table or table cell space. In addition to the background-repeat: no-repeat" CSS attributes, I-Player offers an additional table background mode called "background-repeat: nine-cut". This feature allows the table (or table cell) to use a dynamic size box-type background instead of a fixed-colour or tiled background.

A background box is in fact an image made of nine small tiles, corresponding to the splicing of an image into three columns and three rows. When drawing the background of a nine-cut table or table cell, I-Player smartly uses the tiles to "explode" the box and make it fit to the table or cell size. To use a spliced image as a table "box" background:

- Splice a "box" image into 9 images numbered from 1 to 9 and each one corresponding to its square number in the 3x3 grid
- The splicing must be done in such a way that the 4 corners are fully contained by tiles 1, 3, 7 and 9, the 2 horizontal borders (tiles 2 and 8) can be repeated horizontally, the 2 vertical borders (tiles 4 and 6) can be repeated vertically and the central tile (5) can be repeated vertically and horizontally
- Add padding as needed (for a TABLE, add the padding in the <TBODY>)

Change the default colour of the focus frame

Use the spot-colour CSS attribute for the BODY in order to define the colour that the browser uses to highlight the active spot. The default colour is #14CC14 (lime green).

```
BODY {
   spot-color: blue;
}
```

V HOW TO ADD DYNAMIC ACTIONS

V-1 *Use ACCESSKEY attribute to handle colour keys*

The ACCESSKEY attribute can be used to specify shortcut keys links or buttons. The value of the ACCESSKEY attribute is a predefined access key name, and can be {RED}, {GREEN}, {YELLOW} or {BLUE}. For instance:

```
<A HREF="http://www.netgem.com/" ACCESSKEY="{RED}">Press RED now</A>
```

V-2 *Use key handlers to handle any key*

You can set key and event handlers to handle key presses. The special I-Player "nativeToName" function allows you to convert key codes into key names for more readable code. For instance, you can handle the colour keys like that:

```
<HTML>
<HEAD>
<SCRIPT>
function handleKey()
{
  if (typeof event.nativeToName != "function")
    return;
  var keyName = event.nativeToName(event.nativeKey);

  var colour = null;
  switch (keyName) {
  case "KEY_TV_RED":
    colour = "red";
    break;
  case "KEY_TV_GREEN":
    colour = "green";
    break;
  case "KEY_TV_YELLOW":
    colour = "yellow";
    break;
  case "KEY_TV_BLUE":
    colour = "blue";
    break;
  }
  if (colour)
    document.body.style.backgroundColor = colour;
  document.getElementById("key").innerText = keyName;
}
</SCRIPT>
</HEAD>
<BODY BGCOLOR="white" onKeyPress="handleKey()">
<TABLE WIDTH=100% HEIGHT=100%>
<TR>
  <TD VALIGN=middle ALIGN=center ID=message>Press a colour key</TD>
</TR>
<TR>
  <TD VALIGN=middle ALIGN=center ID=key> </TD>
</TR>
</TABLE>
</BODY>
</HTML>
```