



Part One**Video, Audio, Data***Compression**Interactive vs. Enhanced**Transmission**Receiving Data**Muxing**Fetching Data*

Part Two**The Graphic Layer***OSD Graphics**Screen Areas**Distortion**Always Use a TV-set*

Part Three**Palette, Fonts, Graphic Primitives***colour By Numbers**colour Look-Up Tables**Fonts**Graphics**Conversions*

Part Four**BRAINS Navigation***Navigation Principles**Icons**Choices**Metaphors**Usability*

Part Five**BRAINS Production***Production Basics**Rationalizing**Housekeeping**Standards**Clarify*

Part One Video, Audio, Data – How does it work?

Compression.

The digital signal allows for compression and transmission of data alongside the video and audio. Compression is achieved through the use of the MPEG format. In analogue transmission, each frame is transmitted in full PAL, while the MPEG format is an adjustable compression format. In short, full PAL frames are only sent once in a while and the remaining image information being transmitted is only the difference from one frame to the other.

Image compression is the most important factor for transmitting more channels in the same bandwidth as analogue transmission, while digitalisation of audio has more effect on quality of the sound. The extra data transmission is what makes Interactive Services - or rather - Enhanced Television possible.

Interactive TV vs. enhanced TV.

There's a big difference between i-TV and e-TV: For a broadcast channel to be interactive, it takes a return path, which is

usually a standard telephone line. Even though people call a regular TV Guide interactive, it's really only a program loaded into the ram of the Set Top Box (STB). The user can flip and browse through information, but it is all in the box at one given time. When the user tunes to the channel carrying the TV Guide, all the information is being downloaded, and it will be lost again, once the user tunes away.

Another pseudo-interactive service is the "Present-Following" information, which is available on all digital channels. This information is generated and updated all the time and is sent out along with all digitally broadcast channels. Services like these are more E-TV than I-TV because the user cannot really interact with new information, or gather more information than is already in the signal.

A chat-service, a two-way e-mail or a home-banking service with transaction possibilities would be truly interactive.

Transmission.

The way a typical STB receives information

is comparable to traditional teletext services. Inside the STB is an MPEG decoder, which starts to convert digital data to images, graphics and sound for the television screen and speakers. Tuning can take longer than on regular analogue channel switching because of this process. Even the fastest chips are still only computers, which have to transform information. The sound usually comes through first - in a matter of milliseconds. The first frames of the video are displayed once the chip receives the first full-frame image - normally in less than 3 seconds. But the data is only available once all of the information is in the box. Programming wisely can bring up graphics on the screen even before all the data is present, but in order to have a fully functioning service, of course all the information has to be present in ram.

Receiving data.

To calculate how long time this takes, one has to know how much ram is being used in the box for data, and the transmission speed of the signal (how much bandwidth

is reserved for data). If the application takes up 500 kilobytes and the stream is transmitted at 512 kilobit/s, it would normally take a little over 7 seconds, but this also depends on which sequence the data is received. Just like when searching for a teletext page, the box has to get the information in the right order before the whole application is ready to use.

On some boxes, if the transmission speed is set too high, the chip simply isn't able to build an application because it gets overloaded with information input. Likewise, on the video: If the compression rate is set too high, the chip will not be able to discern the differences between full screen images, and the result will be chunks or blocks in the image and unsteady image quality. Even audio can lose out information, like when a CD jumps a few notes on a dirty disc.

Muxing.

The streaming of all this information is what is referred to as multiplexing - video and audio is being digitized and mixed (or

mux'ed) together with data. All three elements of the digital signal take up less than one regular analogue signal - typically between 2 to 6 Mb/s. This is why a digital transmission can carry more channels in general, and services in particular, than the "old" analogue systems.

The final signal can be transmitted through cable, air or satellite. The end-user of course needs some form of receiving hardware in order to view the services. Either a television set containing MPEG decoding facilities or a STB, which converts the digital signals to analogue. In some systems, the modem in the STB may be a cable modem, which means that the user does not have to connect the regular phone line to the STB, but will simply use the cable to send signals back into the system. This is more convenient to the user, and typically cable modems communicate on a much higher bit-rate than phone-line modems.

An example is cable modem (512 Kb/s) compared to phone-modems (14.4 or 28.8 Kb/s). The true transmission speed also

depends on how many users are on the system at one time - just like regular Internet connections.

Fetching data.

The data-channel in digital transmissions can fetch information from just about anywhere. A feed from the head-end, where computers are converting tv listings into a data format suited for the TV Guide application or the Present/Following banner, or small systems set up for gathering information from other sources, such as meteorological sites, databases containing any kind of information and even e-mail systems. The important part is conversion of raw data into the format needed for the application running in the STB. If the system only gives access to custom made information (i.e. not full Internet access), it's called a "Walled Garden" setup.

Part Two The Graphic Layer

OSD Graphics.

In order to create a visual interface for television, something the user can interact with, the graphic layer is used to display text, icons and other elements overlaying the video. The graphic chip in the STB is what draws the On Screen Display (OSD graphics), and that is why the complexity of the OSD depends on the graphic abilities of the chip.

The interactive application has to be custom designed to the STB receiving the application - especially when it comes to graphic display considerations. Of course the application also has to comply to other standards, i.e. amount of ram, processor-type and operating system & version. When it comes to the graphics, however, the most important concept to understand has to do with the way the graphics appear "on top of the background" so to speak.

The background is normally an MPEG image, which can be the broadcast signal (video), it can be a full-PAL still image in the MPEG format or it can be drawn on the screen using only OSD graphic elements, in

which case the video layer is empty or blank.

The big difference between a full background image and the OSD is that the background is always created in the matrix of 720x576 pixels in the PAL format, and the OSD is placed in the matrix of these boundaries. Counting from the upper left corner ($x=0$, $y=0$) to the lower right hand corner, ($x=720$, $y=576$). A graphic element placed in this matrix can either be a rectangular bmp-file or it can be displayed by the chip using coordinates and fill-colours. Depending on the complexity of the graphic needed, it is up to the designer and programmer to decide whether to have the chip draw the OSD or to include a bmp in the application. Needless to say, it is more efficient to have the chip draw a one-colour rectangle, and more efficient to include a complex image (like a human face) as a bmp-file in the application.

Non-rectangular elements are designed using the transparency feature, which is always colour-position 0 in the palette, exactly as in Internet graphics - or when

using clipping paths in traditional graphic production.

Screen areas.

Depending on the software in the STB, the size of the available OSD layer can vary: When drawing OSD on a live video signal, the first versions of OpenTV only allow using 70% of the screen area for graphics. The second version of OpenTV (EN) allow around 30% cover when using 256 colours and full screen coverage in 16 colour palettes. Until some future versions of software, the only way to use 24bit colours is including the image in the background MPEG.

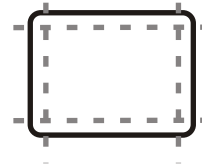
When displaying OSD near the edges of the tv screen, the "screen safe area" has to be considered for the layout. On most tv screens there is an area of 40 pixels (from top down, from left, bottom and right towards the middle) which is outside the visible area.

Text and important information should always be kept at least 45-50 pixels away

from the edges. If some elements need to "bleed" off the edges, of course the elements are drawn all the way to the last pixel.

The OSD area can be set to three different standards for any application. The 70% area has a "top" version, in which the graphics are centered horizontally and extend from the upper line of pixels down to line number 504. Second version of this area is fully centered, and the third version is of course set to the bottom part of the screen.

When drawing OSD on a still image (non-moving video) the whole screen can be used. When using 256 colour palette OSD, the maximum display area is 320,000 square pixels in OpenTV EN, while the further development of software may bring us larger areas.



Distortion.

Because of the non-rectangular size of pixels on a tv screen, a horizontal distortion of graphics takes place when displaying OSD. A perfect circle will become an oval, which is 12% wider than what's expected. One way to adjust for this distortion is to resize all elements to 89% width before exporting to the STB environment. Another work-around would be to create all graphics in the screen size of 809x576 pixels and resizing to 720x576 as a final step. Whichever way, it is always better to resize down than up because of the replacing of pixels.

Horizontal lines should never be thinner than 3 pixels (height). Due to the interlacing of the tv screen, any line of only one or two pixels width will flicker. For the same reason, avoid serif-fonts, but more on these issues later..

Always use a TV-set

The PAL signal tends to soften graphics - even in the graphic layer, which is more precise than the MPEG layer. Due to this

fact, some joining colours seem to blur into each other. On the other hand, high contrasts sometimes look like they have an edge between and it is always a good idea to test graphics directly on a tv screen hooked up to the pc on which the creation process is done.

Semi-transparency (in OpenTV 1.1) is very difficult to achieve. It can be created by making a checkered matrix, where every other pixel is transparent and every other is opaque, but for most colours, the image in the video layer will be mis-coloured due to the way the CRT lights up the pixels on the screen.

All OSD graphics are always non-antialiased, and sometimes it takes meticulous work to find the right pixels to fill out when displaying small graphics. When resizing graphics for OSD it is necessary to check for misplaced pixels and test on a television screen for the final result. Especially logos and icons in which the smallest errors stand out clearly.

Part Three Color by numbers

Colour by numbers.

The first colour (number zero in the palette) is always transparent in the STB environment.

Any application should have its own palette, and it is wise to always have the same basic colours in the same places for different palettes. Almost all applications need black and white, and these should be placed as colour number 1 and 2. White should not be "full blast" - but in the RGB values of 220-220-220. Likewise, other colours should never be full value on any base colour. Especially red/orange tend to flicker and blur if there is too much power in the red channel. Blue and green are not as difficult, but it often looks better to use softer colours.

Colour Look-Up tables.

When designing 16-colour applications, the palette should be decided on at an early stage, so all graphics are saved with the same palette. The chip in the STB does not display the graphics like an ordinary graphic program, but has a look-up table

where each pixel is associated with a palette entry. In addition to the colours used for text (usually black or white) the colours from number 3 to 15 in the 16-colour palette should be used for navigational graphics (arrows etc) and icons.

For interactive designs, it is often necessary to use highlight and subdue versions of some basic colours in order to visualise the focus point of the navigation. Two to three variations of a basic colour make room for variations of buttons, icons and text to show different versions. Focus, selected, passive etc.

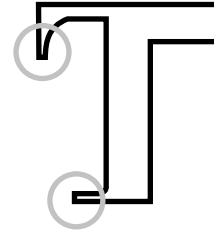
Fonts.

Fonts take up ram just like images, and for one font in one size and one weight, the application spends as much memory as for one full screen image. OSD fonts are bitmapped and non-antialiased. This means that small sizes and serif fonts are not suitable for television. Usually an 18-point font (created in Photoshop) is the smallest readable. It is about the same as subtitle

texts and that's proven to work fine. The combination of colours and text is important: coloured text is always more difficult to read than white or black - with the limitation mentioned before in mind: Not using pure white, and most of the time, it is a good idea to use a very dark grey than pure black as well. In general, beware of any other coloured text. Because of the relatively thin lines in letters, blurring and contrast artifacts take place and this makes the text hard to focus on.

When creating artwork in Photoshop and any other design program on a pc, the length of a text string shown on a PAL sized image will always differ from the final result being drawn by the chip in the STB. This is in part because of the non-square pixels of the tv screen, and is not a big problem, but something to be aware of.

A standard STB comes with embedded fonts, but it is possible to convert any TrueType or Type1 font to a STB font set. One program to use is the FontMonger (commercially available). More on conversion processes later.



Graphics.

Regular buttons, rectangles and graphic primitives, such as circles, triangles and one-colour shapes can be hard-coded and they can be included in applications as bmp-files. The most rational way of creating graphics is to use bmp-files, because they are easier replaced if, for some reason, during the development phase, colour- or size-changes are needed. It is natural to keep control of each graphic element in the hands of the designer rather than the programmer.

Converting fonts is only a small part of making any graphic element into a STB element. When creating bitmaps, it is necessary to resize before saving into the bmp-file format. When creating graphics to a 4-bit environment, all files for one

application have to use the same palette - with the colours mapped correctly to each colour number in the palette.

Conversions.

In Photoshop, it is possible to map a file or a series of files to one specific palette and subsequently save a 4-bit bmp in the Windows format. Afterwards each file goes through another conversion process in the programming phase to compress file size further.

If several files have to be made ready from different formats, and perhaps even different colour depths, it is easier to use DeBabelizer from Equilibrium, which is capable of reading and writing most any kinds of file types. In DeBabelizer, it is also possible to rearrange palette entries a lot easier than in Photoshop.

When backgrounds and large images are converted to the MPEG format, a certain loss of quality is unavoidable. To assure best result, it is recommended to slightly blur sharp areas of the original image prior to saving in the bmp format. Always keep

the original file to try out several versions (and when working in Photoshop, keep the layers separate!).

The final result as seen on a tv screen looks very different than what is displayed on a regular pc screen.

Part Four BRAINS navigation

Navigation Principles.

Focus points are probably the most important elements in designing interactive interfaces. As the user does not have a pointing device, it has to be evident where the "cursor" is at any time.

The easy and natural way to achieve this is by using colour changes of the buttons/text on screen. Navigation by pressing the numerical and/or coloured keys on the remote control makes colour changes on the navigation elements less necessary, but is not recommended as the norm. There is no room for visual feedback, and it is basically less fun to use than when changes occur at every interaction. When the user navigates by pressing arrow keys instead, the immediate feedback assures that the system is alive and responding. Arguments such as faster navigation by using numerical keys don't really work, because of the limitations a natural overview of choices bring. When the user has to choose between more options, it should not be more than 5-8 different, unless the options are in the form of a list, in which case numerical keys will not work

anyhow. When less than 9 options are visible, clever positioning will mean fast access even when using arrow keys.

Icons.

Whether using icons or text (or a combination) for navigation, these elements have to be designed in two versions; one highlighted (for select-state) and one subdued (for not-selected). It is also possible to have a third version, which lights up once the user presses the "OK" key, in order to give even more feedback, but if the system reacts promptly, this last state is not really needed.

Icons can be extremely difficult to design, for more than just one reason:

- 1) Different age groups/gender/culture assign different functions to icons.*
- 2) Icons have to be simple (small sizes, few colours).*
- 3) Icons can't easily be changed after they are once introduced.*

Choices.

When designing a well functioning interface, it is equally important to know early in the process which kinds of choices the user is presented to. Some choices will be displayed as pop-ups, some as lists, others as icons or images - yet others as buttons or hidden-lists . There are even more ways to activate the choices made by the user: Direct manipulation, choose-and-save or simply switch to another scene. Combinations are possible depending on the function of choices made.

The hierarchy of an interactive service must be clear, for the user and for the designer. It has to be clear to the user how to move around in the service, and how to get back to the beginning.

Most users are a little afraid to loose their orientation or to actually "harm" the STB - like a computer needing a reboot.

Metaphors.

One way of looking at interactive services is the "flip&browse" metaphor, where the screen is a window through which the user

sees a page at a time of a book. Flipping through pages is natural for anyone, and using the right/left arrows feels like a normal navigation. When going "up" a level, like to the previous choice, it will be by pressing the up arrow, unless this is used in the actual page to move between sub-choices.

Most remote controls have specially assigned keys, like a "Return" or "Back" key, which would be the simple solution, but this has to work throughout the whole system, otherwise it confuses.



Usability.

Key words concerning navigation and usability:

Minimize "travel", "depth" and "redundancy".

Clarity on access:

To a new topic or to new material in the same topic?

Remove "obstacles" on the screen.

Minimize effort and don't expect users to learn.

Give visual feedback.

Be explicit, flexible and forgiving.

Include help on the screen (graphical and/or textual).

Be consistent.

Use metaphors only if they help.

Limit information.

Example of a BRAINS interface:

*Main menu and sub-levels
colour consistency, highlight/subdue*



Part Five BRAINS production

Production basics.

Once the first few services have been designed in an interactive universe, there is very good reason to keep some basics looking and working the same way throughout other services.

If on-screen arrows are designed as triangles in a particular size, it would be strange to suddenly change their form or even size. If buttons are rounded in the corners, rectangular or any other form, it would be natural to use the same form in similar functions. If lists are displayed directly on the background or in their own boxes, do it the same way other places.

colour changes from one service to the next is actually enough to show the difference - just like the choice of font means a lot for the overall design and look & feel of a service.

If a new service has totally different content than what's already up and running, of course the layout and design has to take new access- and display-options into account. Example: A TV Guide displays one form of information, while a

typical board game is very different. Likewise, a TV Guide is similar in function to a look-up service, even if one has search options and the other does not.

Rationalizing.

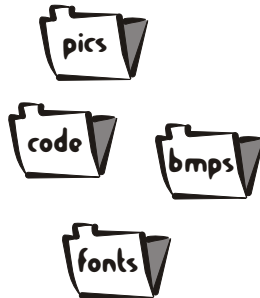
Libraries of all common elements, fonts and colour tables should be categorized and saved for re-use. It is so much easier to adjust a well working set of tools and elements than to invent new stuff all the time.

Whenever a new service has to be designed, always look through work done before and get a thorough feeling for which elements can and should be used again - it means even more to the user than to the efficiency of the developers.

Housekeeping.

These libraries should contain programming code as well as graphic elements: Search functions, input procedures and inter-hierarchy navigation to mention a few. Most services basically do the same

thing for the user: Display choices and info. When it comes to games, this consistency is somewhat different, but at least some functions for games are the same; such as how and where to find rules, options and sub-functions.



Standards.

Any interactive service will always be present in a system with other interactive services, and should be similar in navigation and functionality to the rest. The success of a service depends on content, but the ease of use for the user depends on the interface design. Compare to other GUIs, like that of telephones, Windows or cars for that matter. Once a

common standard is introduced, it is easy to move into another part of the same universe. It is very easy to design complex services and tempting to include smart functions, once the designer has a good overview of a service, but the user does not have this overview before, or after (!) using the service.

Clarify.

Links between different services should be made clear to the user. It is often difficult to backtrack in interactive television unlike when navigating on the Internet (or reading two books at a time).

Two separate services, such as a travel/ticket-service and a weather service may seem natural to link, but as the STB will only hold one service in memory at a time, it means that it is not possible to (re)load a service at any given point. If the same information is needed in two services (ex: the temperature for a Greek island) it is better to display it in each service than to cross-link. This way the user has quick access, doesn't need to load

a new service, and will not be lost in the total hierarchy of the system.

When loading a service, it is possible to include a splash-screen (like starting a Windows program). This allows for control of the load-time, so the whole service is completely loaded before the user starts navigating around in it. Splash-screens are included in each application or can reside in the STB (in that case, there is only one for all services).

In the main menu of an interactive system, the different services available should be categorized in accordance to content. Usually there will be a set of services having to do with regular television watching - such as the TV Guide, access to PPV, etc. Other services could include news and events, games, a general setup section and different types of look-up services. This whole concept is entirely new to the average user and should be simple to use.

For almost 40 years the television set has been used to switch channels and control the volume. These two functions have immediate effect and feedback, but in the

digital universe, even channel switching takes longer time. That's why all interactions (inputs from the user) need to result in some form of visual feedback from the service.