Connected TV Applications for TiVo

Project Jigsaw

Design Draft

26 Feb 2013

Overview

The goal of Project Jigsaw is to create a library of Flash components that will allow VMDS to publish TiVo apps quickly and efficiently, and be able to report the user activity of the app back to the client.

Once the components are built, assembly of new apps will start with confirming what media a client wants to publish. Content falls under 3 categories:

1) Text blocks
2) Photos
3) Videos

A text block could be part of a product page or a screen listing Terms and Conditions.

Each app must have at least one Content Display Screen, which would show either text, photos, or videos. Jigsaw apps have 3 kinds of screens:

1) Support Screens / Pop-Ups
2) Content Selection Screens
3) Content Display Screens

Support screens could be loading or splash screens.

A set of linked screens constitutes an app.

The structure and size of an app is determined by the amount of content the app needs to display. The more content, the more selection screens are required to keep the content orgainised and accessable.

A small app (S) might use only one Content Display Screen and a splash screen as a Support Screen to have a space for branding.

A medium-sized app (M) would likely have one Content Selection Screen (a Home Screen) linking to a set of Content Display Screens. The first screen is a loading screen with minimal branding, and the Home Screen is fully branded.

A large app (L) might be one that has a lot of content organised into categories, and therefore might need extra selection screens.

Screens are assembled from a variety of widgets:

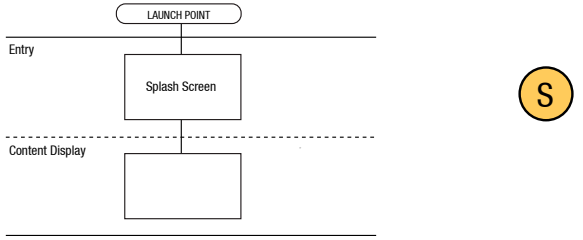
1) Content Display Windows
2) Background graphics
3) Text Buttons
4) Image Buttons
5) Progress Bars
6) Button Hints
7) Text Indicators

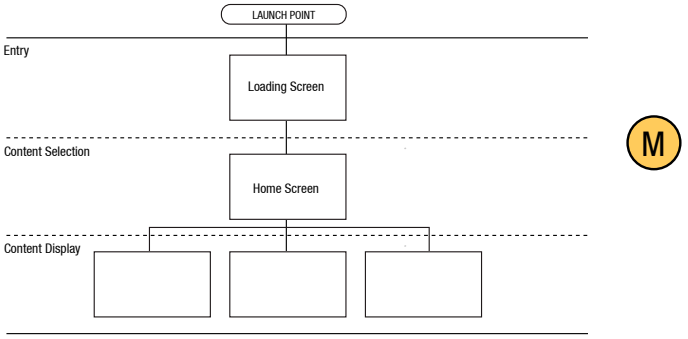
Button groups form these 2 common widgets:

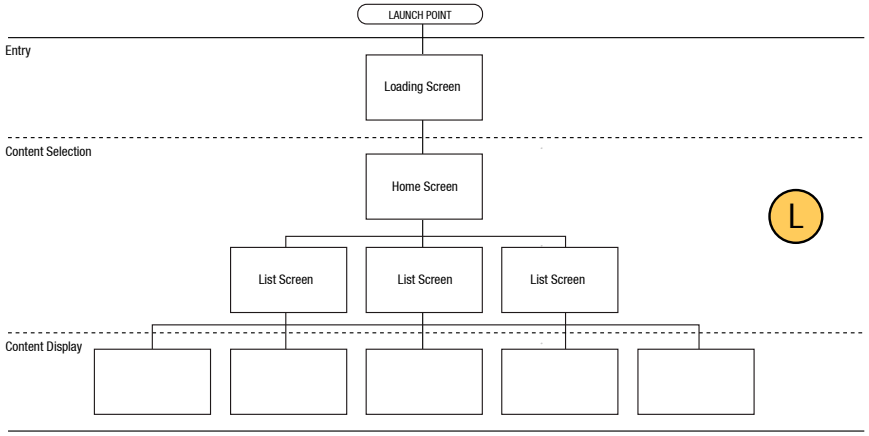
1) Menus: Text Buttons in rows or columns (static and scrolling)

2) Carousels: Image Buttons arranged in rows (static or wrapping)

Widgets assembled in different combinations form the screens needed to accomodate a wide variety of content.







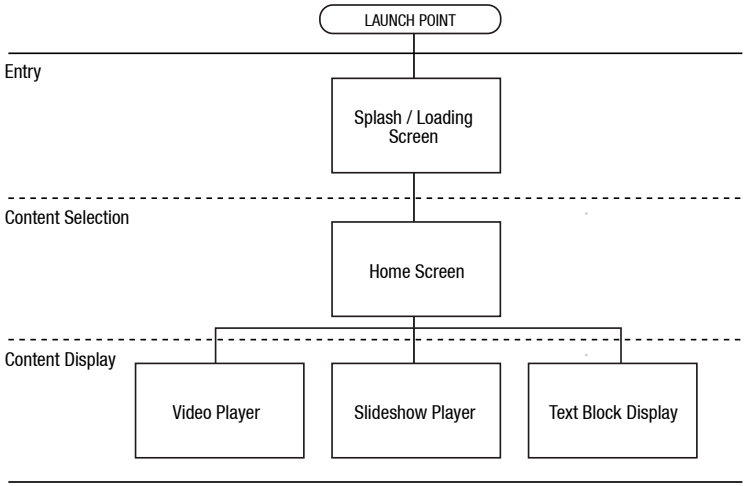
Suggested First Phase

The proceeding pages outline all the variables that should be included in order to create a robust and flexible template that can handle almost any collection of videos, photos, or text layouts on the TV screen. To develop it fully would be a valuable investment, but also admittedly time-consuming. Here's a suggested first phase that will get the most value in the shortest time.

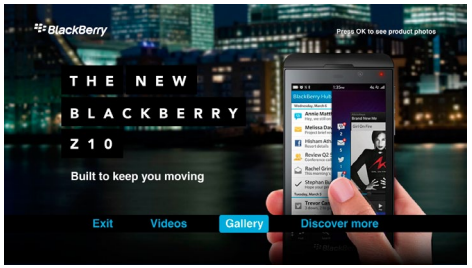
The first phase has a loading/spash screen, a home screen, and 3 content display screens (1 for each content category). Screens can be turned off for smaller apps. For example, if the app only needs to show 5 videos, only the splash screen and video player need to be used (the videos would auto-play, and the video player would have an 'exit' button in place of the usual 'back' button). Another instance would be: if there is a product page (text block display) and 3 videos, the slideshow player would be turned off. It would be efficient if the button on the home screen triggering the slideshow would automatically turn off with it.

Text blocks may first be rolled out as images pre-formatted in Photoshop, but it would be more efficient and desirable if text screens could be completely driven dynamically by the content itself. One strategy might be to have Flash intelligently choose a text screen layout based on the content (eg. if each text block in a series is less than the height of the TV screen and there only are a few pages, it can be displayed as using tabbed navigation. Otherwise, if any of the text blocks are taller than the screen the text can be displayed using paged navigation. See page 13).

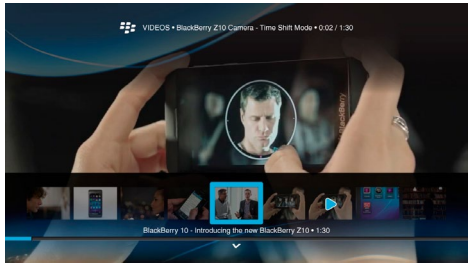
Styling, colours, and UI graphics should be controlled externally from the SWF, so that they can be changed quickly without republishing.



Example loading screen

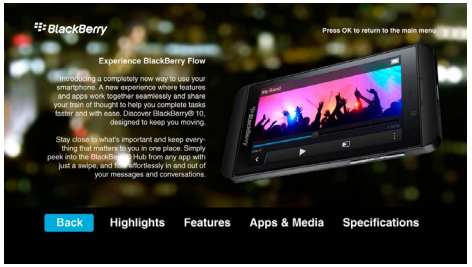


Example home screen



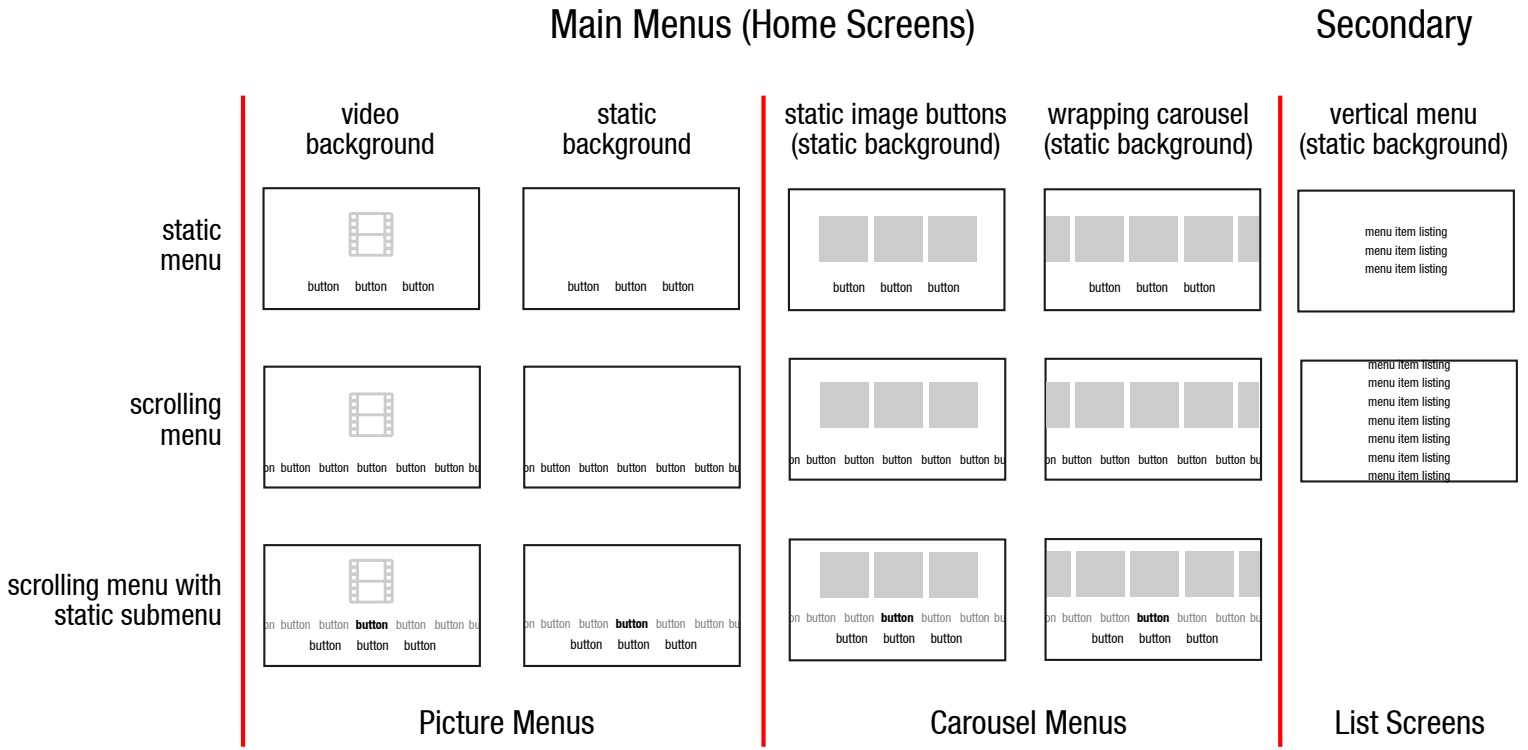
Example video player

Example slideshow player



Example text screen

Content Selection Screens



Content Selection Screens are made of 2 kinds of elements:

1) Backgrounds
2) Buttons

Buttons come in these types:

1) Text Buttons
2) Image Buttons

Image buttons are arranged in rows. Text Buttons can be both:

1) Horizontal
2) Vertical

From these variables we get a variety of button arrangements, such as scrolling menus and carousels.

Backgrounds and buttons can be:

1) Static
2) Moving

For example, a background can be static image or moving as a video.

Button sets are static if they fit on the screen, moving if they don't.

If a row of text buttons, for instance, is not as wide as the screen the menu will be static and the focus will move from button to button, but if the row of text buttons is wider than the screen the menu will scroll and the focus will remain in place. Image buttons that span the width of the screen will form wrapping carousels where the focus remains in the centre.

Menus can also have submenus, in which case the menu moves side to side and the submenu is static.

These combinations form a variety of Picture Menus, Carousel Menus (both used for Home Screens/Main Menus) and List Screens.

Content Display Screens

These screens display 3 kinds of content:

1) Text blocks (that may include inline pictures)
2) Photos
3) Videos

They can display:

1) Single Items
2) Multiple Items

They employ at least one of these:

1) Menu
2) Carousel
3) Link button

Menus and carousels can be used to navigate a screen or to jump to other screens, and link buttons tend to jump to a related screen in the app.

Single Item

Menu

Text Block Display

Single text block with a back button

Photo Display

Single photo with a back button

Video Display

Single video with a back button and transport controls

Live video with a back button

with link

Text Block Display

Single text block with a back button and link button

Photo Display

Single photo with a back button and link button

Video Display

Single video with a back button, transport controls, and link

Multiple Items

Menu

Multiple text pages with tab style navigation

Headline Player

2 states of single screen

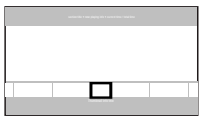
Slideshow Player

2 states of single screen

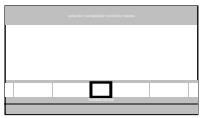
Video Player

Carousel

Multiple text pages with carousel navigation



Slideshow Player, with carousel for slide selection

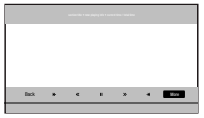


Video Player, with carousel for video selection in a playlist

with link

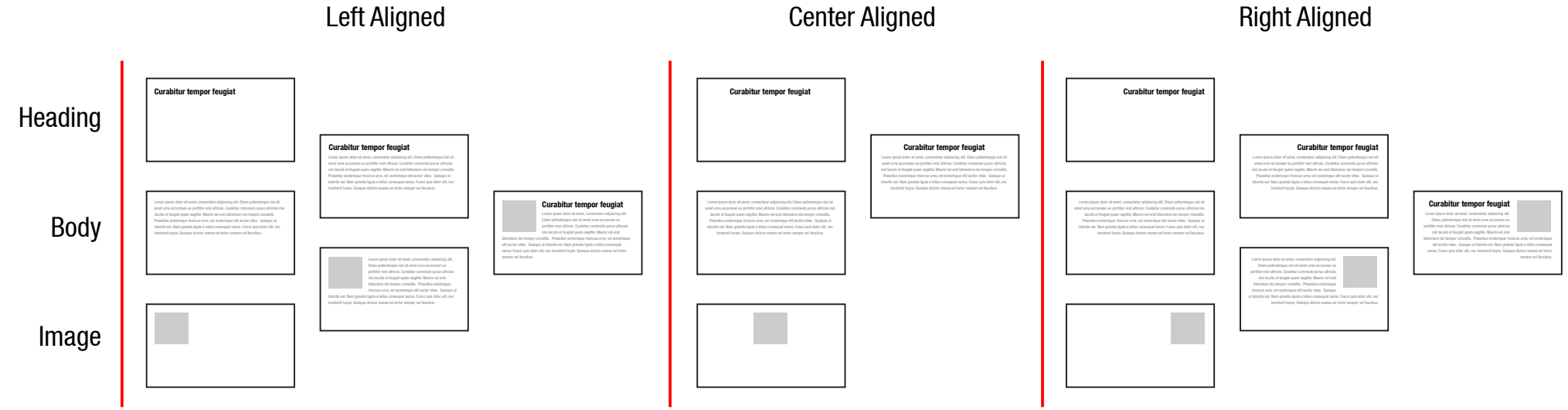
Multi-page text article with link button to other text articles.

Slideshow Player with link button



Video Player with link button

Text Blocks



Text blocks are made of at least one of these elements:

1) Heading
2) Body
3) Image

These elements are set to one of the following alignments:

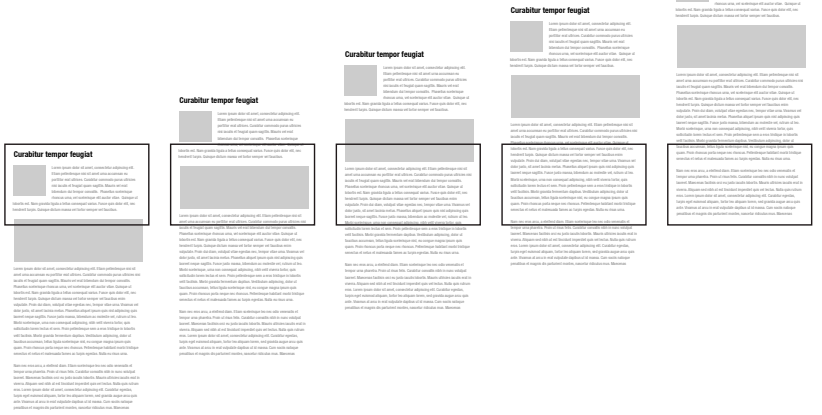
1) Left
2) Centre
3) Right

An image can be any size up to 1280 x 720, and if an image is included the text wraps around it.

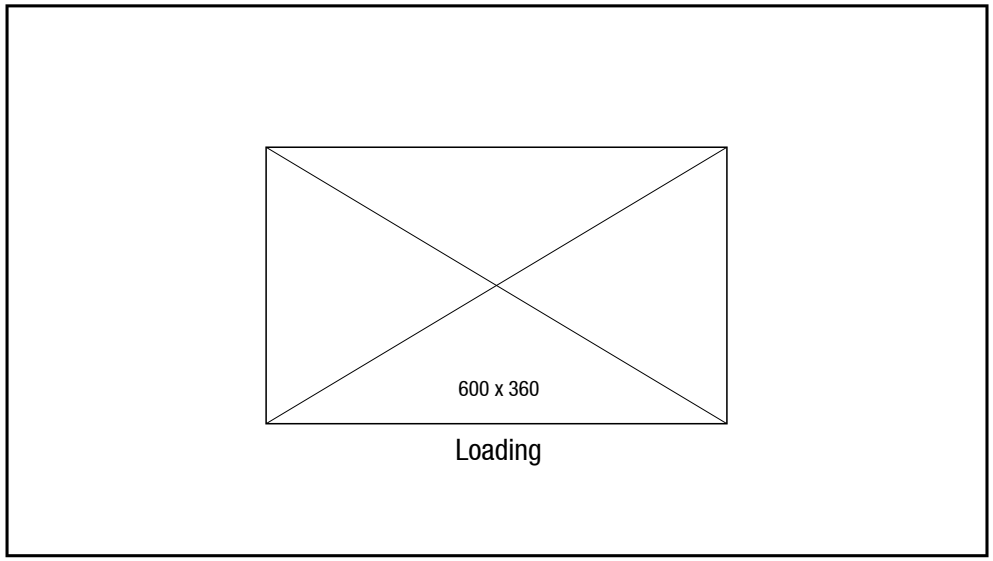
Blocks can be combined in a column to create nearly any kind of magazine style layout. The combined blocks can then be viewed as pages by stepping them along the Y axis a screen height at a time.


Stacked Text Blocks


Text Layout


Paginated

Support Screens

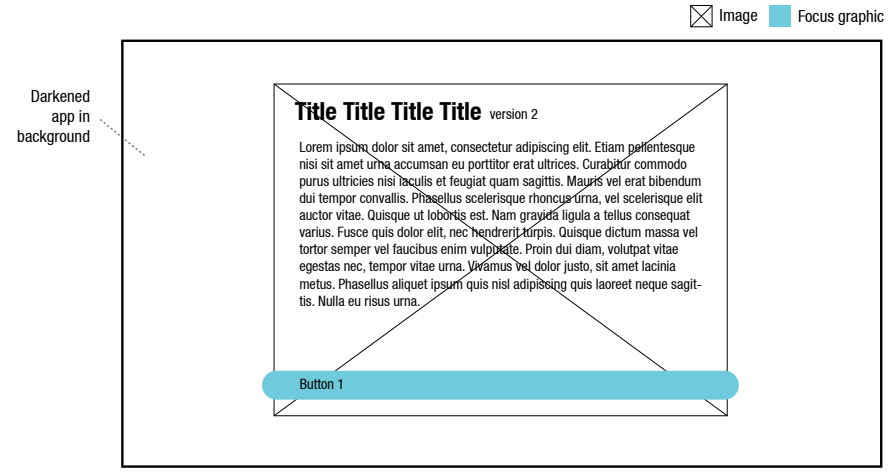


Loading Screen

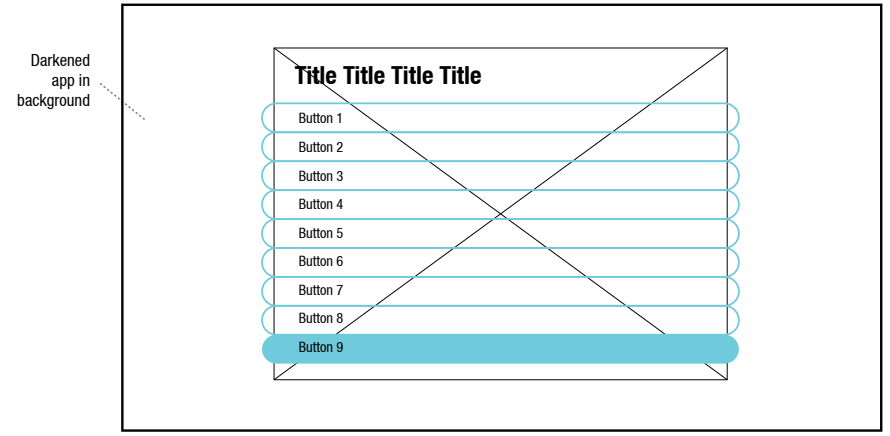
This screen is displayed while the app is loading. It's purpose is to reassure the viewer that loading is in progress, to provide a brief distraction during loading, and to prevent the 'dead air' of a black screen during the load from being mistaken for a malfunciton.

There is always a black screen for a second or two when TiVo first runs Flash, so dead air can't be avoided entirely, but the Loading Screen displays as quickly as possible after the launch of Flash. To ensure a quick display the graphics should be kept to a minimum file size of 150 Kb and displayed on black. Graphics must be transparent PNGs. Logos should have maximum dimensions of 600 x 360, and animations should generally be avoided.

Support Screens



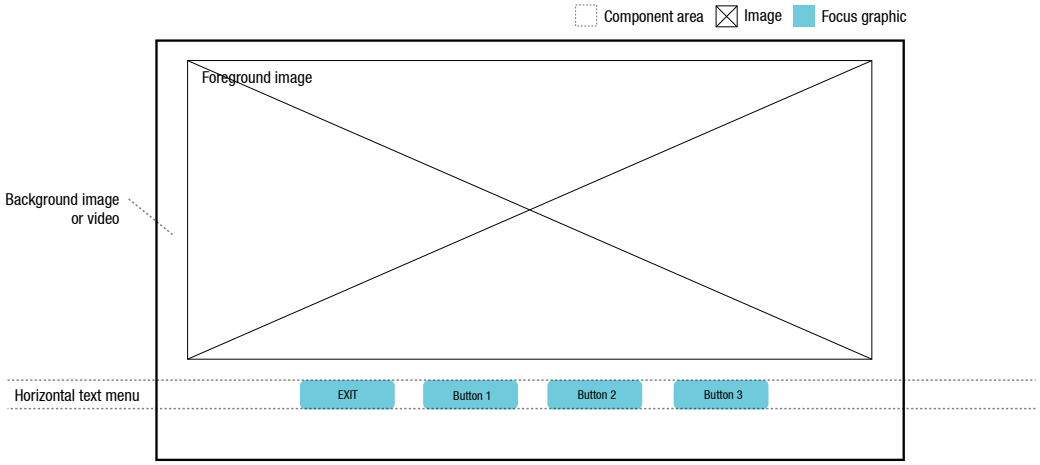
A version of this is needed to display info about the app, triggering from the home screen.



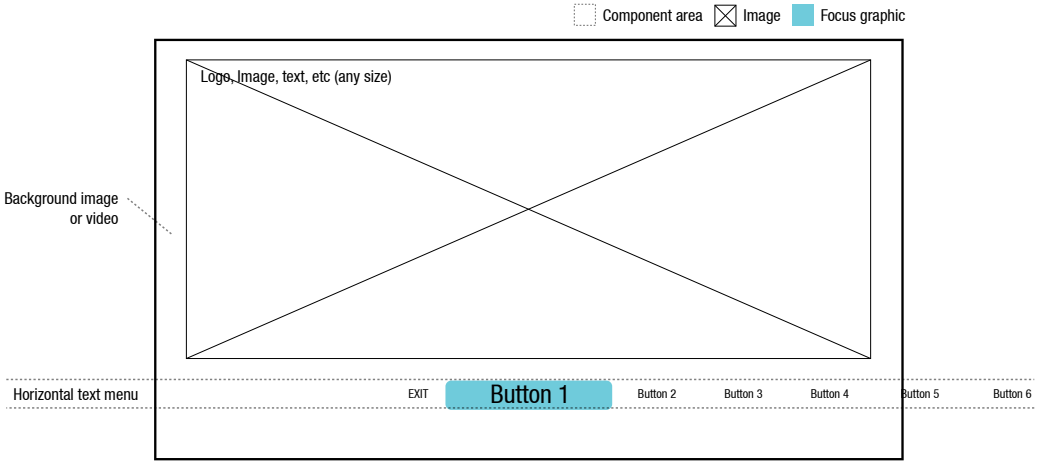
Pop-Ups

Pop-up windows are used if part of a task or message does not fit on the current screen, but also does not warrant an entirely new screen. When activated the focus hides from the parent screen and appears on the first pop-up button, of which there can be a total of 9. The parent screen is also dimmed using a dark overlay so that a clear separation can be seen between the pop-up and the background.

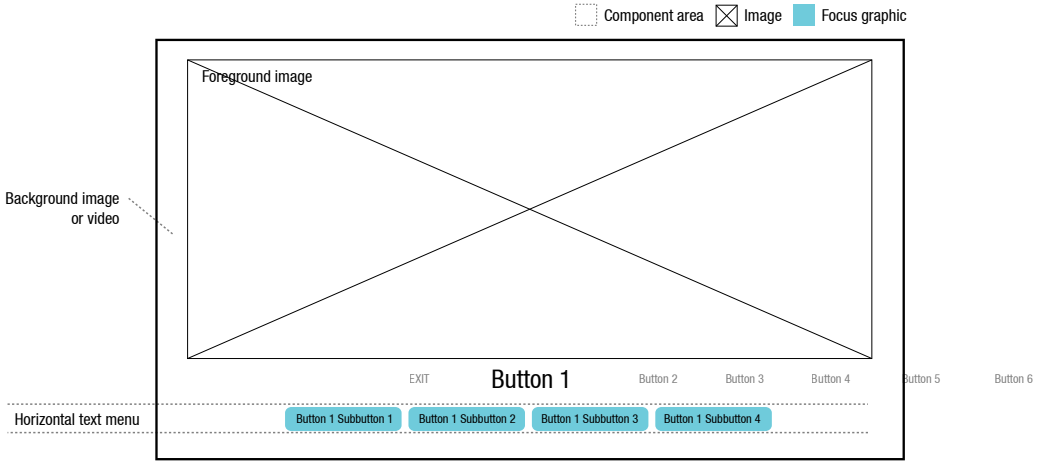
Picture Menus



Picture Menu with static text buttons

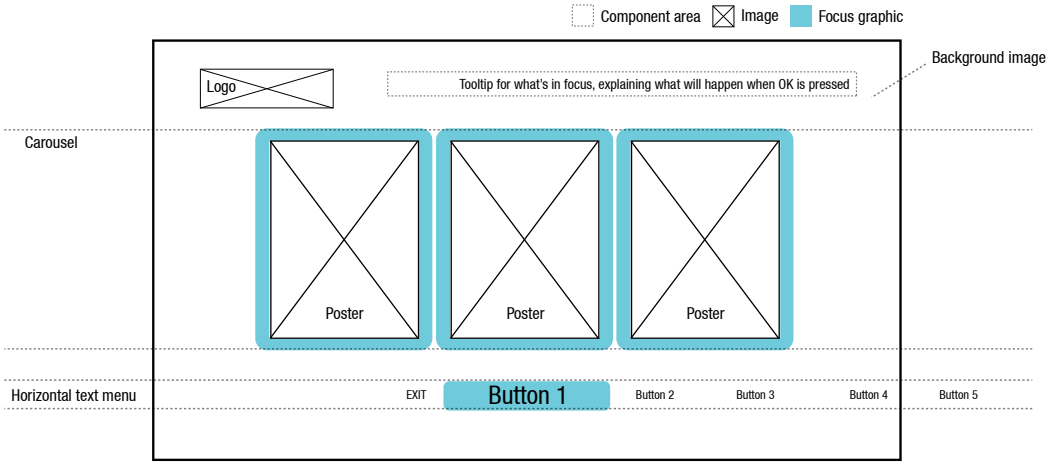


Picture Menu with scrolling text buttons

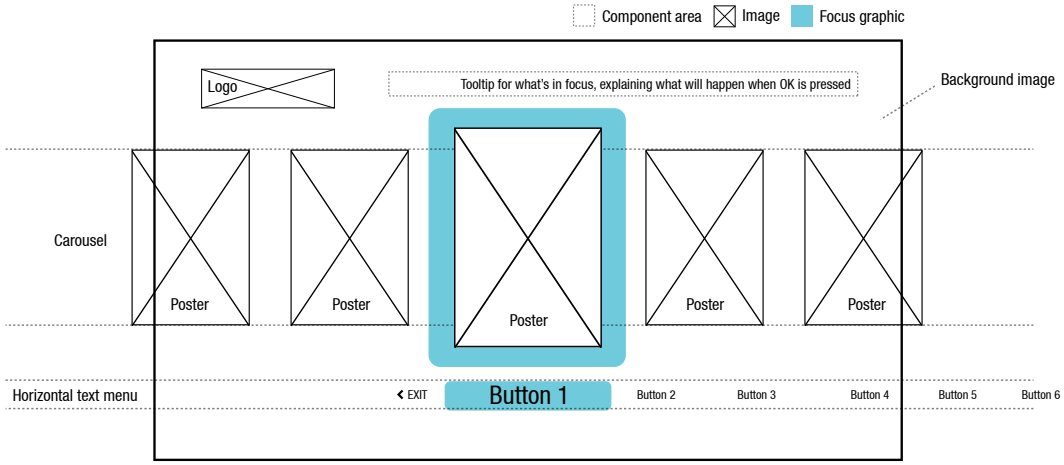


Picture Menu with scrolling menu and static submenu

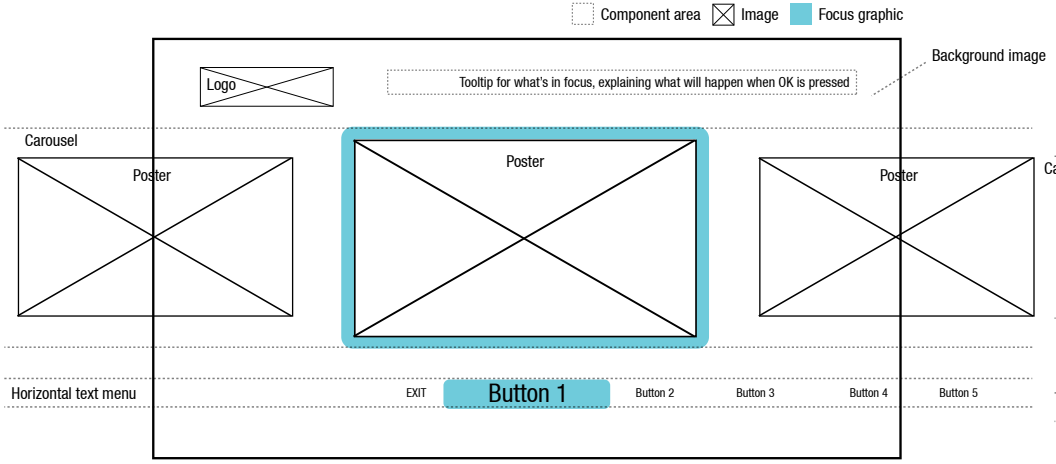
Carousel Menus



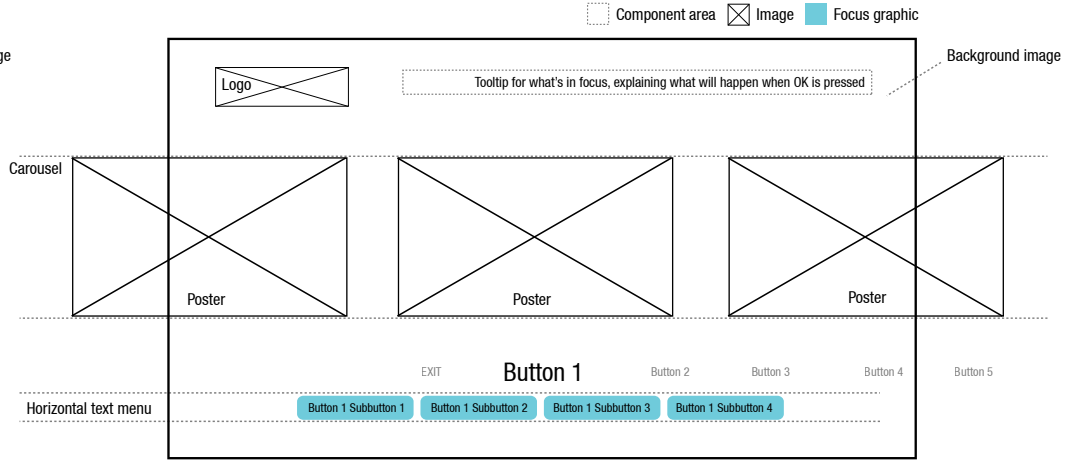
Carousel Menu with static image buttons and scrolling text buttons



Carousel Menu with wrapping carousel

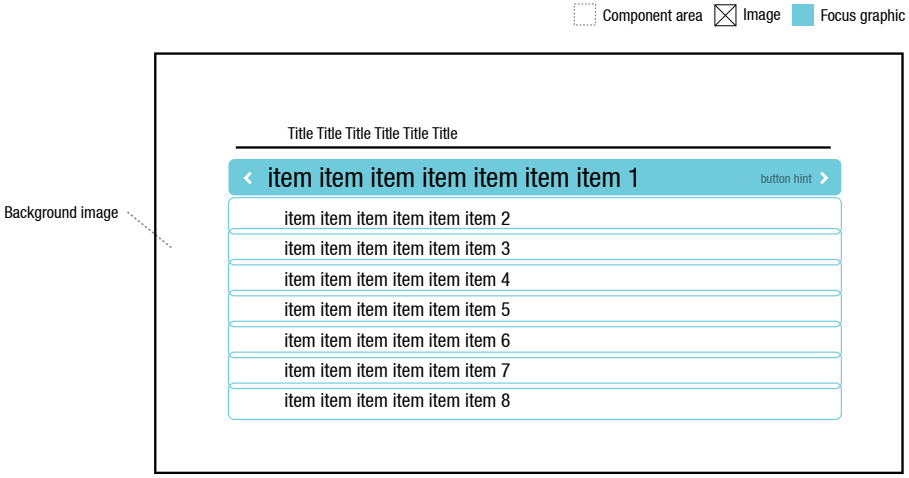


Carousel Menu with large image buttons. The carousel accomodates images of any size, to a maximum of 600 x 340.

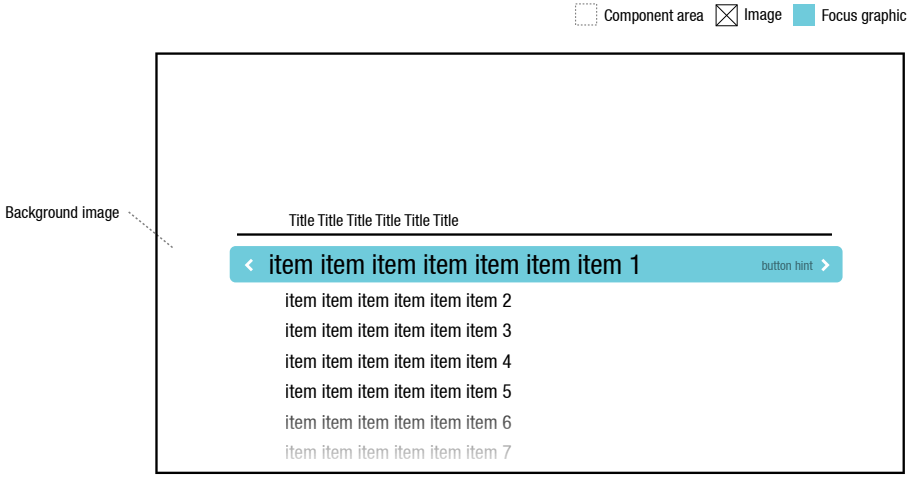


Carousel Menu with focus on submenu

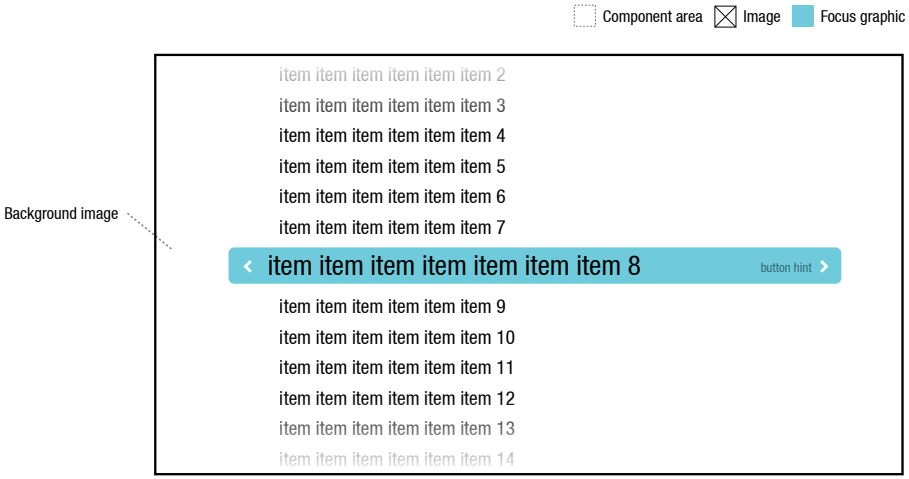
List Menus



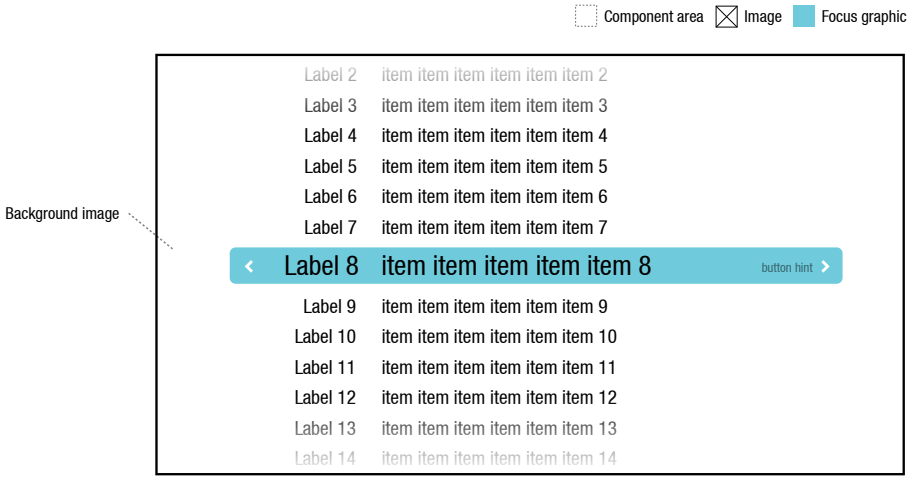
List Menu with static text buttons



Carousel Menu with scrolling text buttons

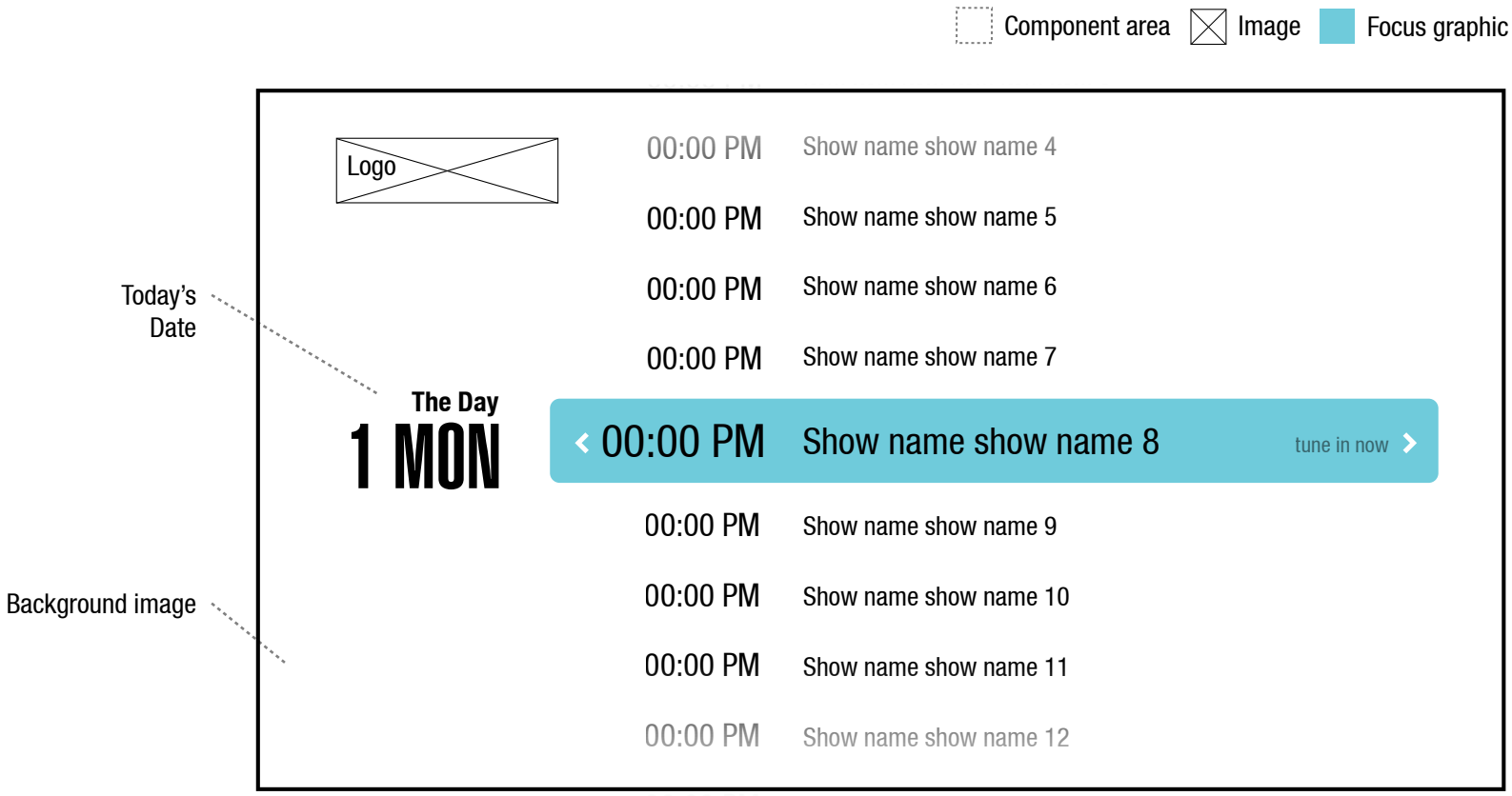


Carousel Menu with scrolling text buttons

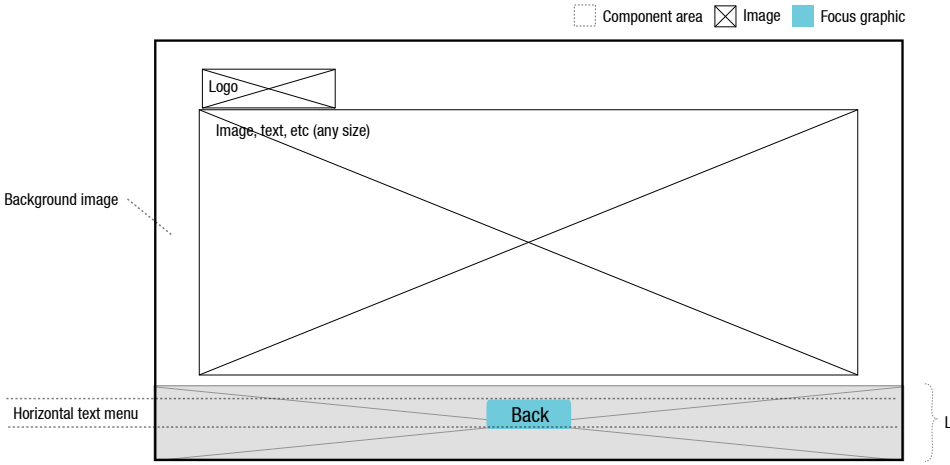


Carousel Menu with scrolling text buttons (2 columns)

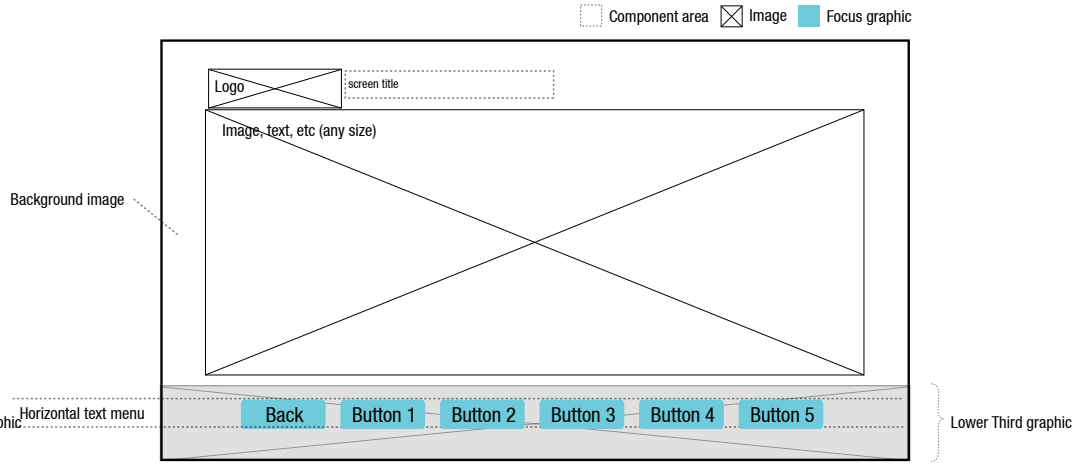
List menu with scrolling text buttons (2 columns) (TV Schedule version)



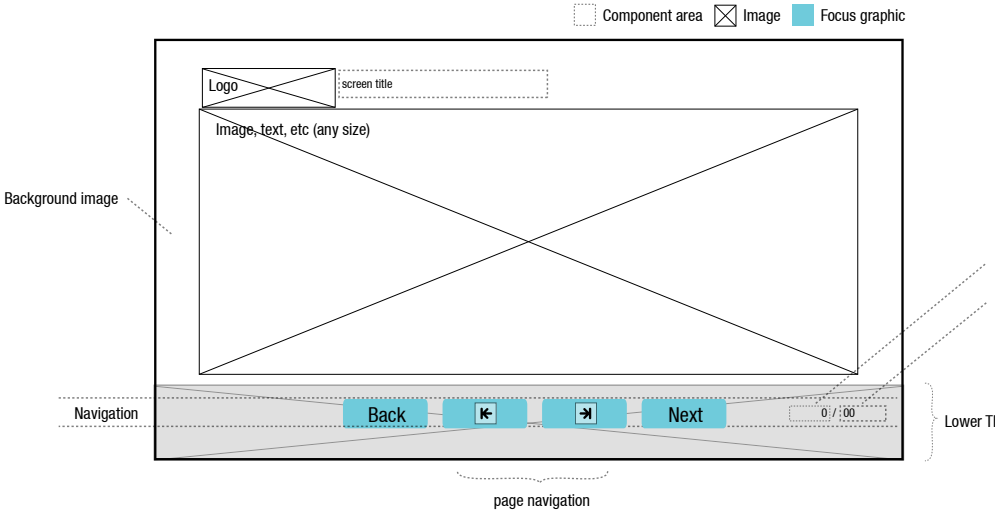
Text Screens



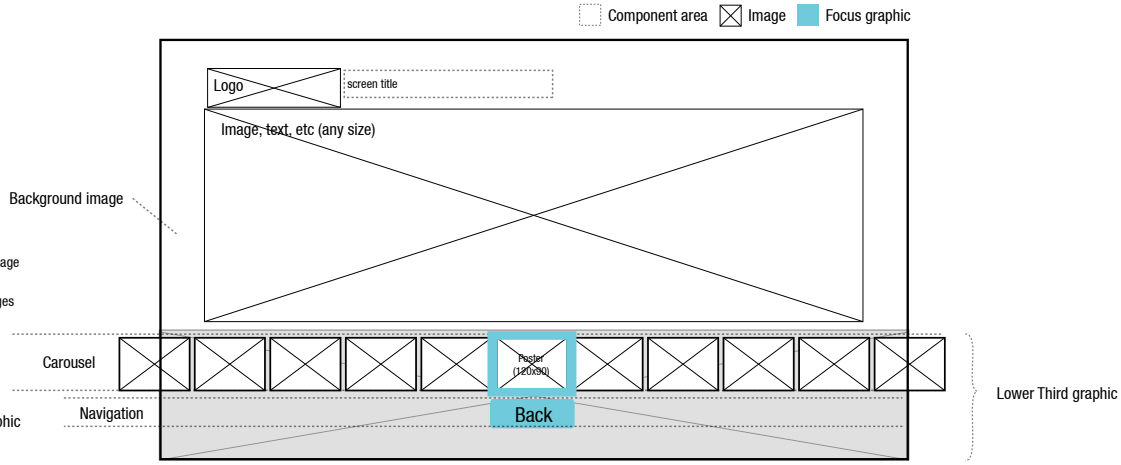
Text Screen, single text block



Text Screen, multiple text blocks, tabbed navigation

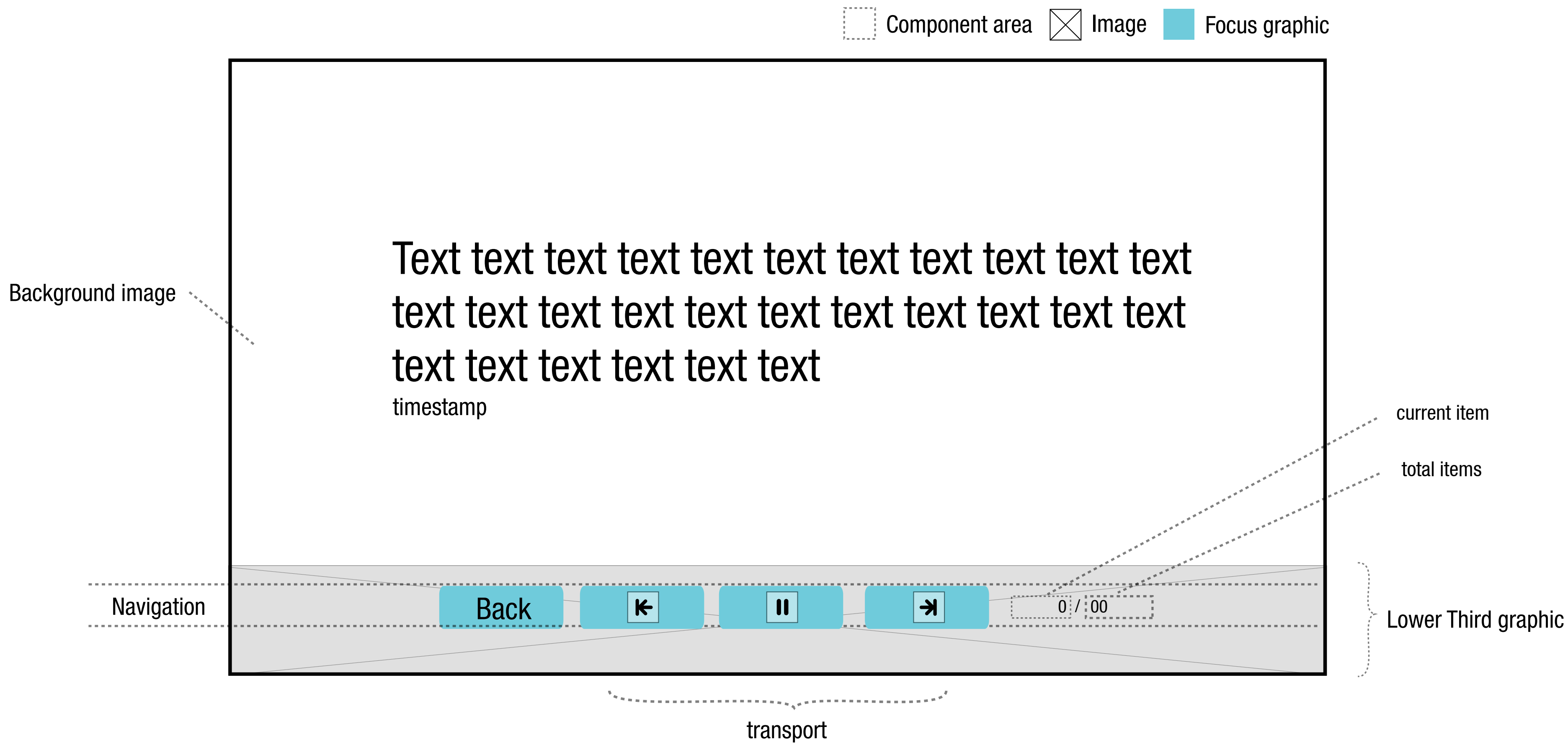


Text Screen, multiple text blocks, paged navigation

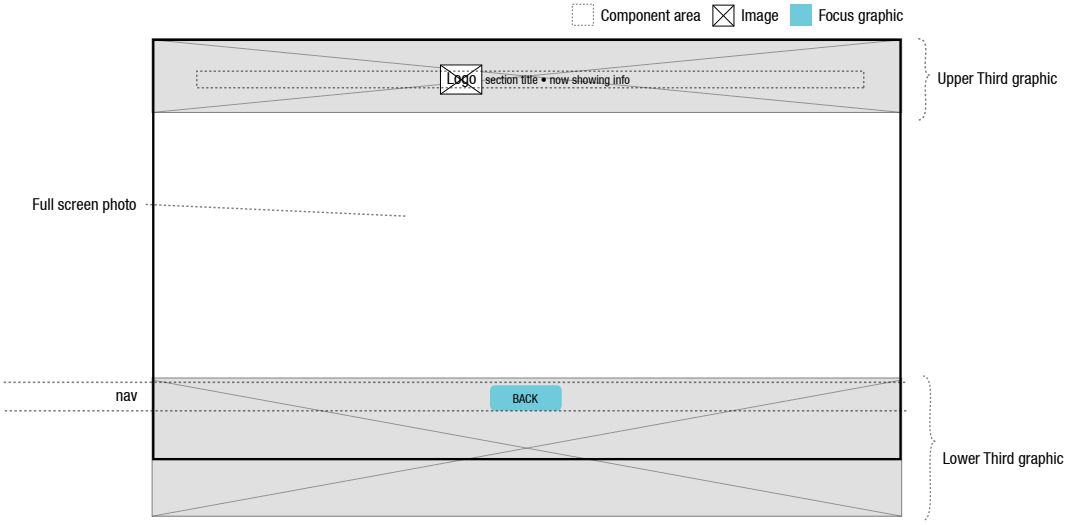


Text Screen, multiple text blocks, carousel navigation

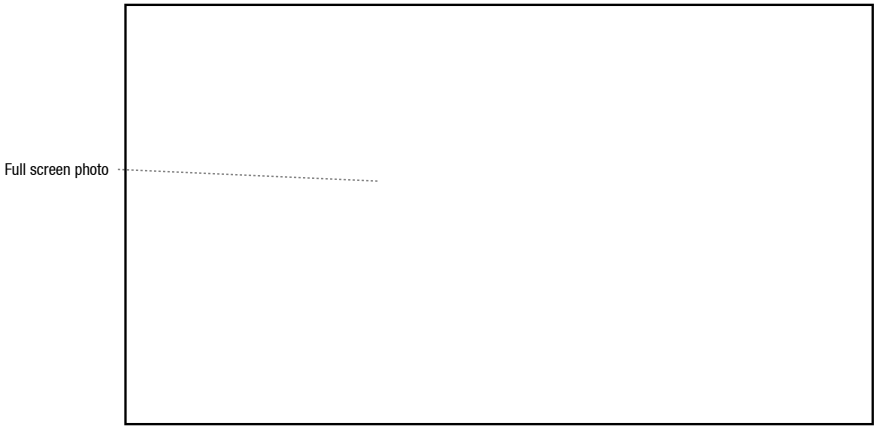
Text Screen player, multiple text blocks, transport navigation (for social feeds like Twitter)



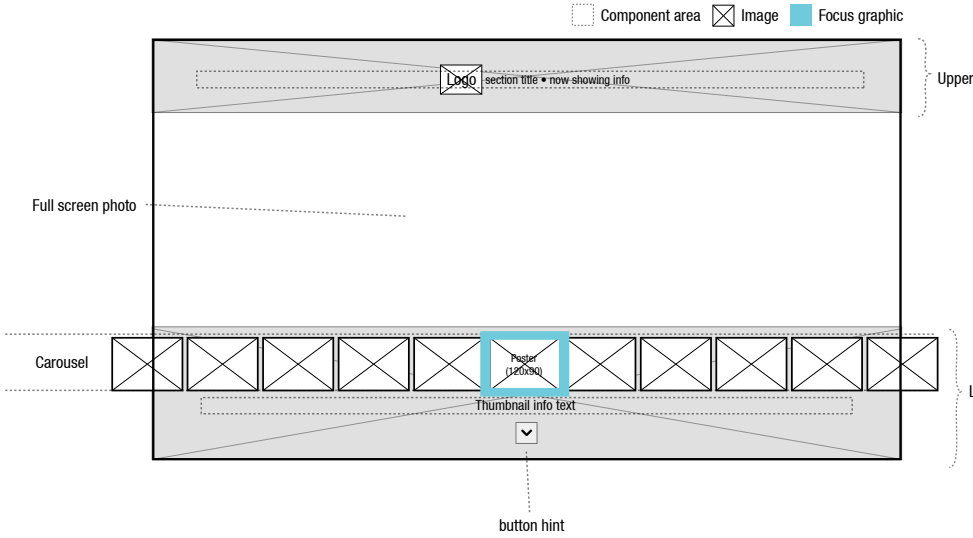
Slide Display Screens



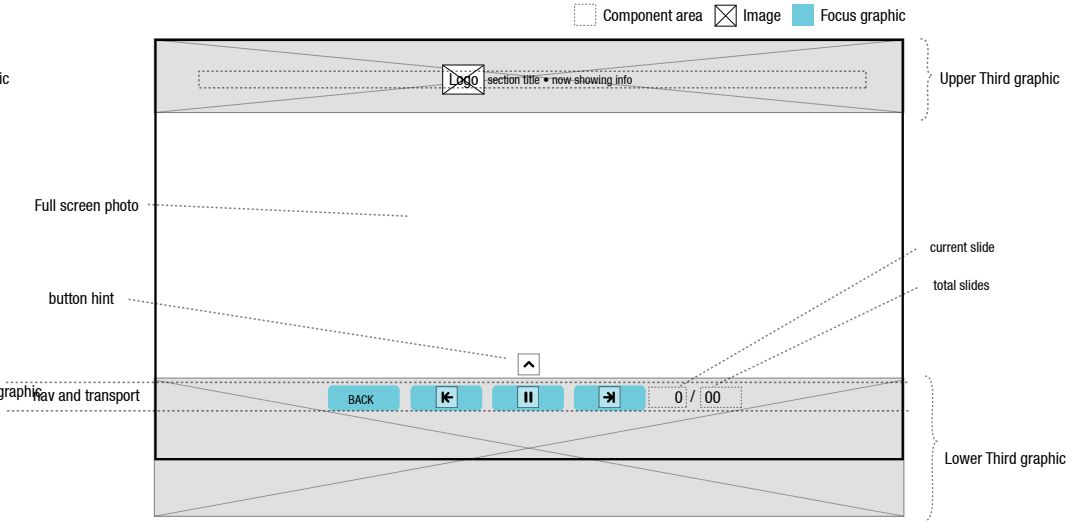
Single slide



Single slide (overlay hidden)

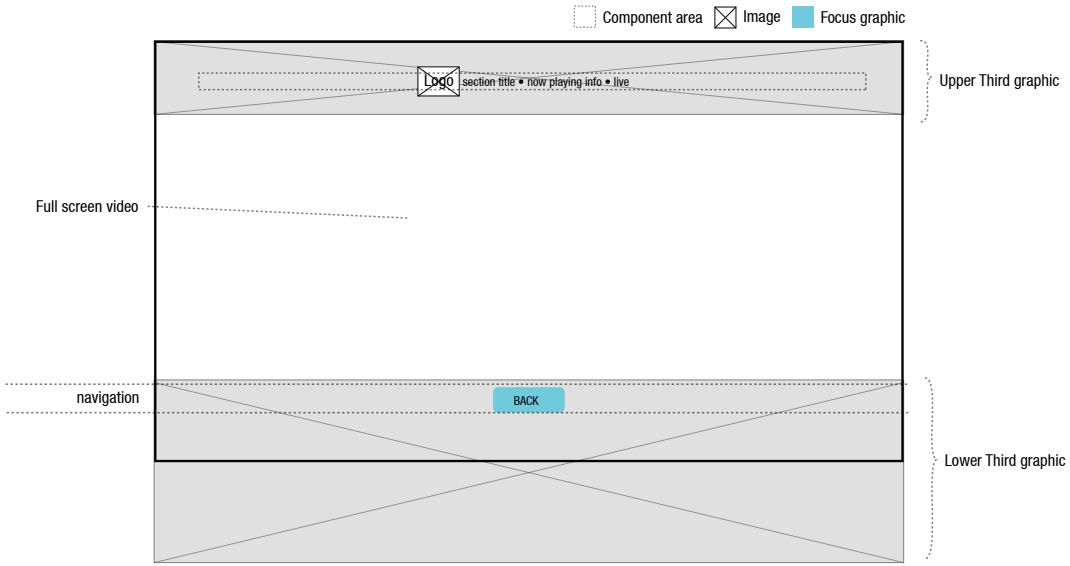


Multiple slides (Slideshow, carousel screen state)

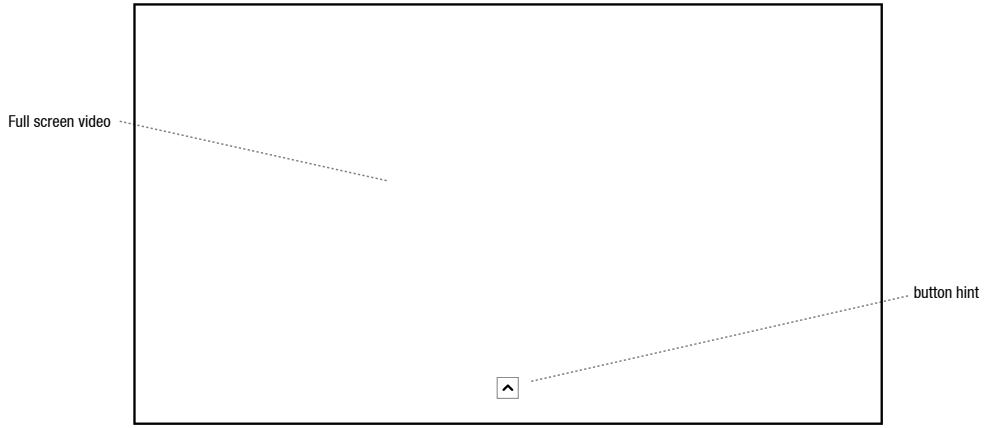


Multiple slides (Slideshow, menu screen state)

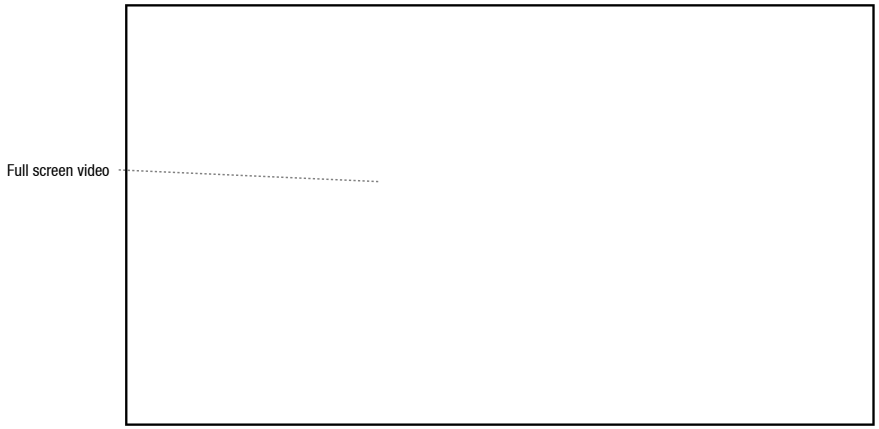
Video Players



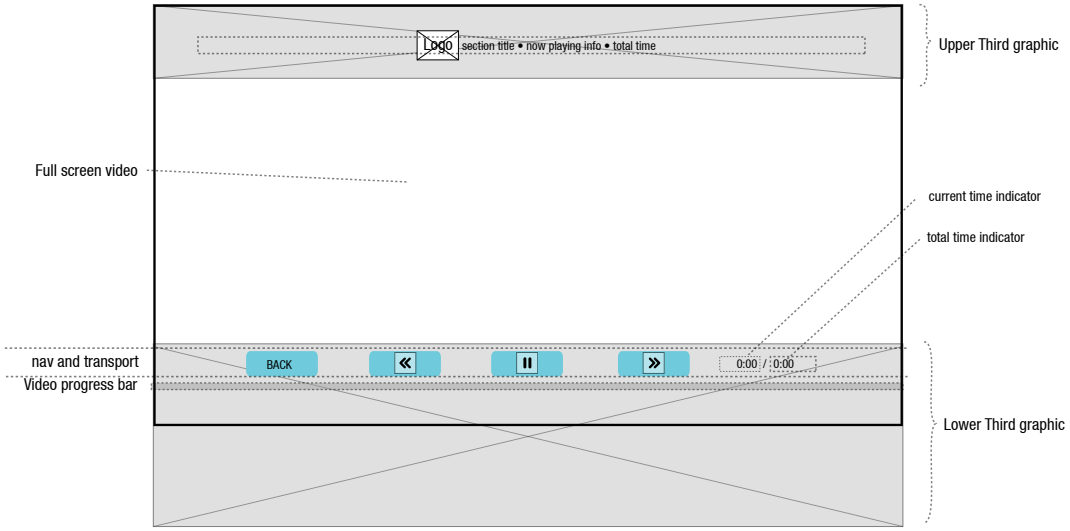
Live video stream display



Video display (Interactive overlays hidden, button hint on)

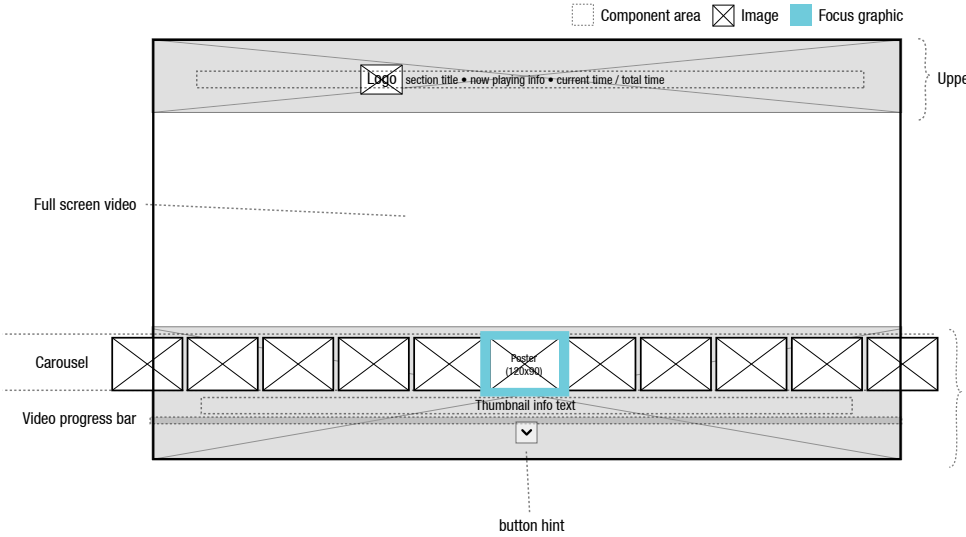


Video display (overlay hidden)

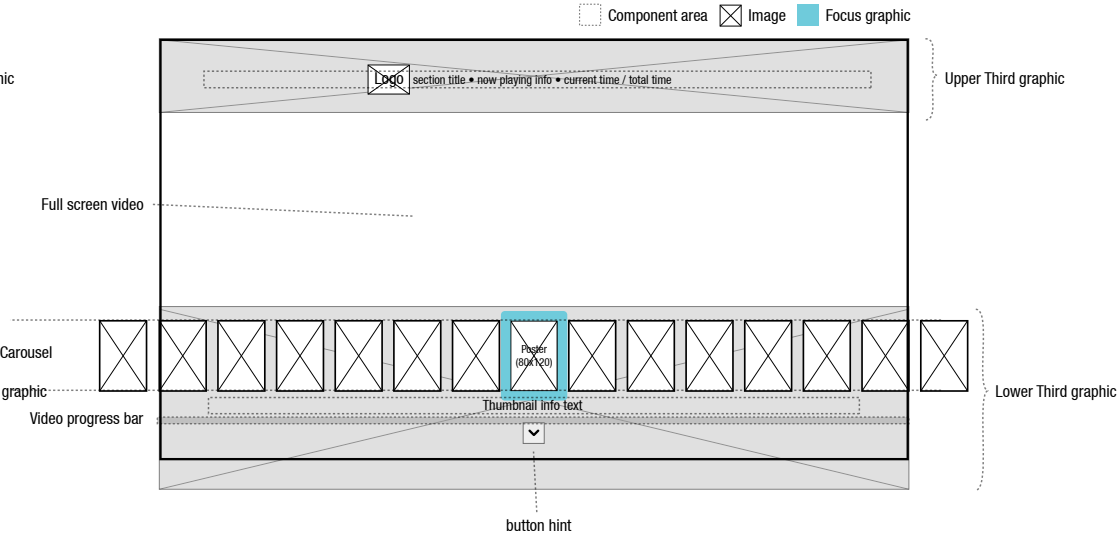


Single video

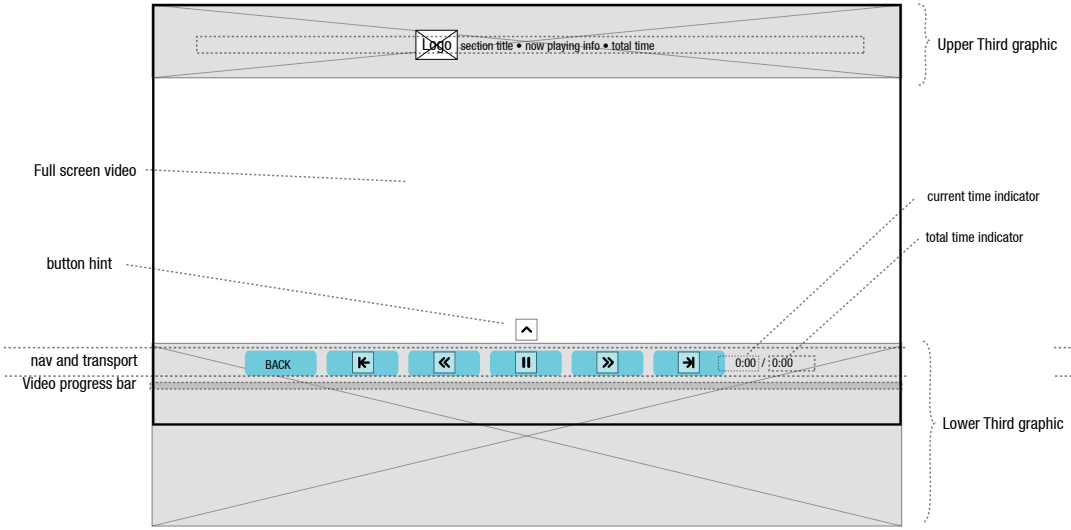
Video Players



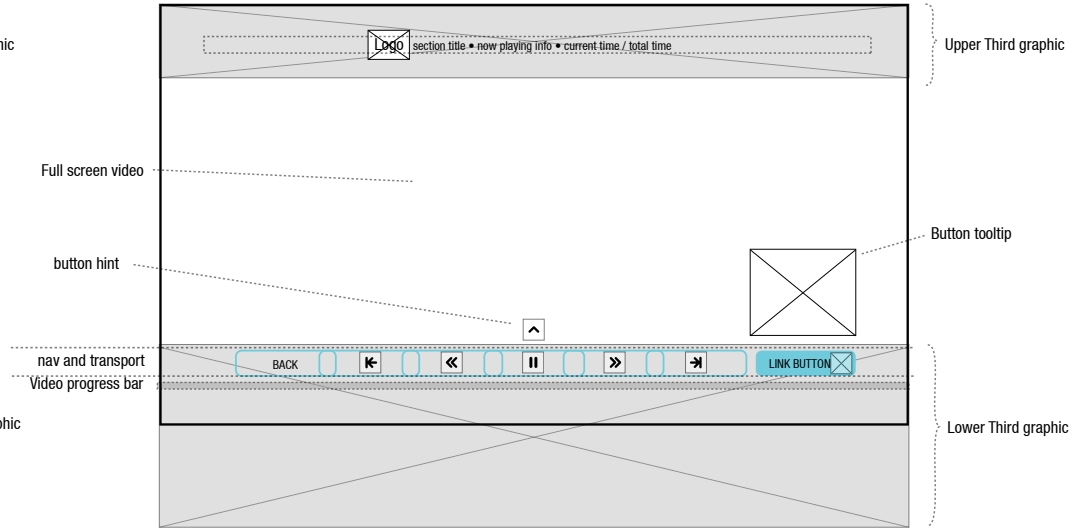
Video player, multiple videos (carousel state)



Video player, multiple videos (carousel state, vertical thumbnails)

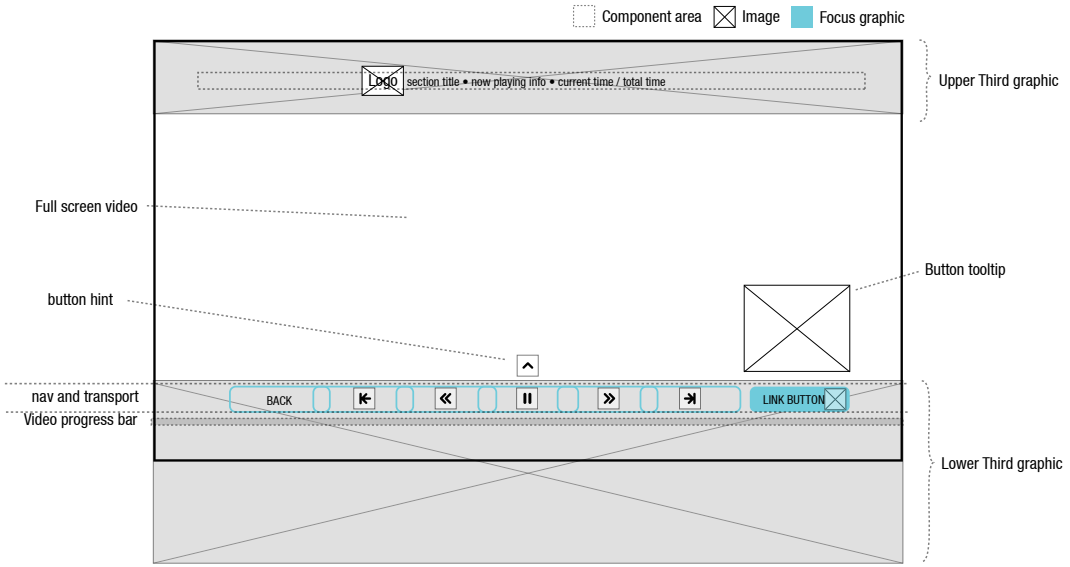


Video player, multiple videos (menu state)

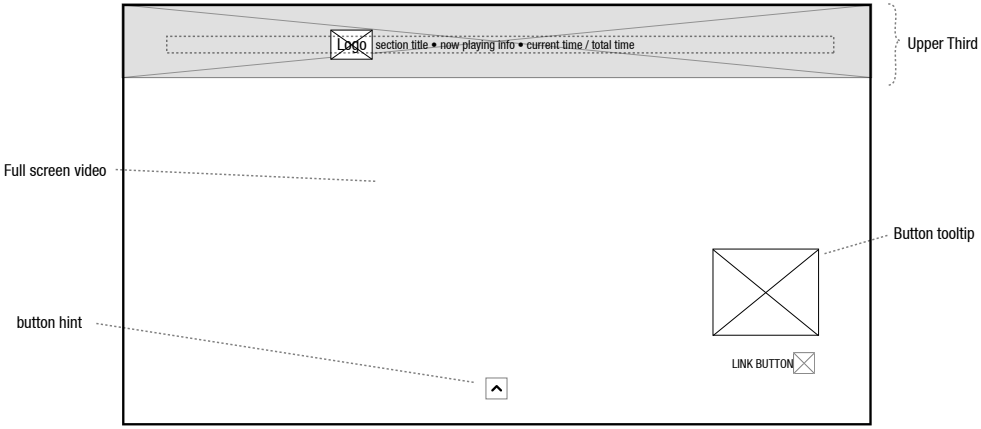


Video player, multiple videos, with link (menu state)

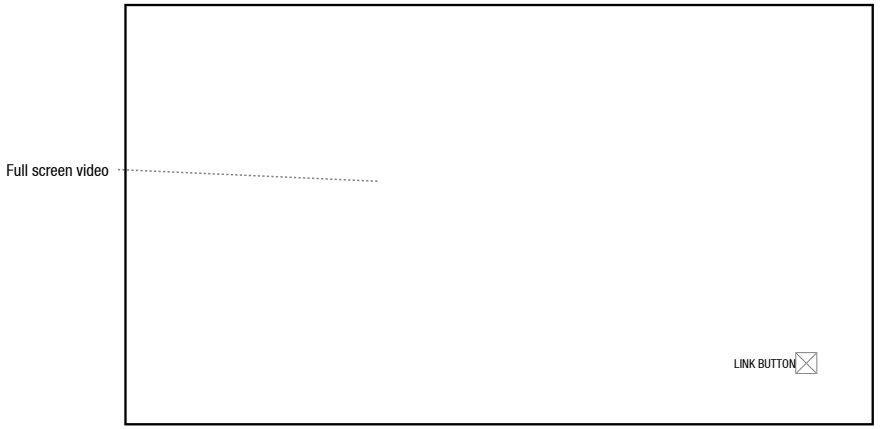
Video Players



Video player, multiple videos, with link (menu state)



Video display (Interactive overlays hidden, button hint on, upper third info bar on)



Video display (overlay hidden, link button hint remains)

Appendix A: Styling

These styling parameters are driven by external XML and CSS files.

Fonts

normal font
title font
emphasis font

normal font colour
title font colour
emphasis font colour

normal font size
title font size
emphasis font size

Text

alignment
image source
image padding
text block size
text block origin

Buttons

normal colour
focus colour
activated colour
inactive colour
active colour
image source (optional)

Focus Graphic

normal colour
activated colour
corner radius (optional)
image source (optional)
padding

UI Graphics

logo image
background image
remote control button hints image
home screen lower third image
text screen lower third image
video lower third image
video upper third image
video now-playing indicator image
list screen upper fade image
list screen lower fade image
green button tooltip graphic

Appendix B: Text Block test

```
/*-------------------------------------------------------------------------------------------------*/
function storeBitmap (imgURL, index)
/*-------------------------------------------------------------------------------------------------*/
{
    bitmapCacheMC.createEmptyMovieClip("bitmap"+index, index);
    with(bitmapCacheMC)
    {
        var loader = new MovieClipLoader();
        var preload = new Object();
        loader.addListener(preload);
        preload.onLoadInit = function()
        {
            var pixelData = new flash.display.BitmapData();
            pixelData.draw();
            bitmapLoaded();
        }
    }
    loader.loadClip(imgURL, bitmapCacheMC["bitmap"+index]);
}

/*-------------------------------------------------------------------------------------------------*/
function loadCachedBitmapIntoMC (BMPcache, movieClip)
/*-------------------------------------------------------------------------------------------------*/
{
    pixelData = new flash.display.BitmapData(BMPcache._width, BMPcache._height, true, 0x00000000);
    pixelData.draw(BMPcache);
    movieClip.createEmptyMovieClip("bitmap", this.getNextHighestDepth());
    movieClip.bitmap.attachBitmap(pixelData, 1, true, true);
}

/*--------*/
function drawTextBlock (ss, textBlockOrigin, textBlockSize, imageSize, imageBlock, pageAlignment, imageURL, heading,
copy)
/*--------*/
{
    var image = "<img src='"+_level0.imageContainer+"' width='"+imageBlock[0]+"' height='"+imageBlock[1]+"'/>";
    image += "vspace='0' hspace='0' align = '"+pageAlignment+"'/>";
    var indentBugFix = "<small> </small>";
    var headerLine = indentBugFix + "<h1>" + heading + "</h1><br/>";
    var bodyCopy = indentBugFix + copy;

    if (imageURL=="" || imageURL==null){image = "";};
    if (heading=="" || heading==null){headerLine = "";};
    if (copy=="" || copy==null){bodyCopy = "";};

    createTextField("tf", getNextHighestDepth(), 0, 0, textBlockSize[0], textBlockSize[1]);
    with (_level0.tf)
    {
        multiline = true;
        autoSize = true;
        wordWrap = true;
        html = true;
        styleSheet = ss;
        htmlText = "<p>" + image + headerLine + bodyCopy + "</p>";
        _x = textBlockOrigin[0];
        _y = textBlockOrigin[1];
    }

    if (pageAlignment == "right"){_level0.imageContainer._x = textBlockOrigin[0]-3+(tf._width-imageSize[0])}
    else if (pageAlignment == "center"){_level0.imageContainer._x = (textBlockOrigin[0]+tf._width/2)-(imageSize[0]/2
)};
}

/*-------------------------------------------------------------------------------------------------*/

var heading = "Lorem ipsum dolor sit amet, consectetur adipiscing elit.";

var copy = "Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam pellentesque nisi sit amet urna accumsan
eu porttitor erat ultrices. Curabitur commodo purus ultricies nisi iaculis et feugiat quam sagittis. Mauris vel erat
bibendum dui tempor convallis. Phasellus scelerisque rhoncus urna, vel scelerisque elit auctor vitae. Quisque ut
lobortis est. Nam gravida ligula a tellus consequat varius. Fusce quis dolor elit, nec hendrerit turpis. Quisque
dictum massa vel tortor semper vel faucibus enim vulputate. Proin dui diam, volutpat vitae egestas nec, tempor vitae
urna. Vivamus vel dolor justo, sit amet lacinia metus. Phasellus aliquet ipsum quis nisl adipiscing quis laoreet
neque sagittis. Fusce justo massa, bibendum ac molestie vel, rutrum ut leo. Morbi scelerisque, urna non consequat
adipiscing, nibh velit viverra tortor, quis sollicitudin lorem lectus et sem. Proin pellentesque sem a eros tristique
 in lobortis velit facilisis.";

var imageURL = "hp04.jpg";

var pageAlignment = "left";
var normalFont = "TivoHelveticaCondensed";
var normalFontSize = 23;
var imagePadding = [normalFontSize*1.3, normalFontSize*0.7]; // horizontal, below
var textBlockOrigin = [120, 100];
var textBlockSize = [1040, 500];
var normalTextColor = "#000000";
var emphasisTextColour = "#cc0000";

var ss = new TextField.StyleSheet();
ss.setStyle("p", {color:normalTextColor, fontSize:normalFontSize, fontFamily:normalFont, textAlign:pageAlignment});
ss.setStyle("h1", {color:emphasisTextColour, fontSize:normalFontSize*1.2, fontWeight:'bold'});
ss.setStyle("small", {fontSize:1});

createEmptyMovieClip("bitmapCacheMC", getNextHighestDepth());
bitmapCacheMC._visible = false;
storeBitmap(imageURL, 100);

if (imageURL == "" || imageURL == null)
{
    drawTextBlock(ss, textBlockOrigin, textBlockSize, imageSize, imageBlock, pageAlignment, imageURL, heading, copy);
}
else
{
    bitmapCacheMC.bitmapLoaded = function()  // bitmap has finished loading, size can now be measured
    {
        var imageSize = [bitmapCacheMC["bitmap100"]._width, bitmapCacheMC["bitmap100"]._height];
        var imageBlock = [imageSize[0]+imagePadding[0], imageSize[1]+imagePadding[1]];

        createEmptyMovieClip("imageContainer", getNextHighestDepth());
        with (_level0.imageContainer)
        {
            lineStyle(4, 0x000000, 100, false, normal, square, miter, 4);
            lineTo(imageSize[0], 0);
            lineTo(imageSize[0], imageSize[1]);
            lineTo(0, imageSize[1]);
            lineTo(0,0);
            _x = textBlockOrigin[0]+5;
            _y = textBlockOrigin[1]+5;
            loadCachedBitmapIntoMC (bitmapCacheMC["bitmap100"], _level0.imageContainer);
        }
        drawTextBlock(ss, textBlockOrigin, textBlockSize, imageSize, imageBlock, pageAlignment, imageURL, heading,
copy);
    }
}
```

An actionscript testing the dynamic image wrapping for the text blocks. Everything has been hardcoded for the test, but the content and the CSS would be put in external files.