



# Curso Multimedia Home Platform 1.1.2

## MHP Graphics II. Widgets, Focus, Input...

UI Widgets

Transparency

Key Inputs

## Curso Multimedia Home Platform 1.1.2

Copyright 2008 © Enrique Pérez Gil

Licensed under the ***Creative Commons Attribution-Non-Commercial-No Derivative Works 3.0 Unported License***. You may not use this file except in compliance with the License. You may obtain a copy of the License at:

<http://creativecommons.org/licenses/by-nc-nd/3.0/legalcode>

This is a human-readable summary of the License applied:

(<http://creativecommons.org/licenses/by-nc-nd/3.0/>)

**You are free to Share**, to copy, distribute and transmit the work **Under the following conditions:**

- **Attribution.** You must attribute the work in the manner specified by the author or licensor (but not in any way that suggests that they endorse you or your use of the work).
- **Noncommercial.** You may not use this work for commercial purposes.
- **No Derivative Works.** You may not alter, transform, or build upon this work.

For any reuse or distribution, you must make clear to others the license terms of this work. Any of the above conditions can be waived if you get permission from the copyright holder. Nothing in this license impairs or restricts the author's moral rights.

## Qué es HAVI

Specification of the Home Audio/Video Interoperability (HAVi) Architecture,  
Chapter 8 – Level 2 User Interface

[www.havi.org](http://www.havi.org)

## Por qué HAVI

- Estamos en entorno distinto al PC, para el cual se diseñó la java.awt, de manera que DVB, siguiendo su línea habitual, decidió adoptar un standard ya pensado para funcionar en “Consumer Devices” como la TV: HAVI, y más en concreto, el subconjunto definido como “Level 2 User Interface”
- Este API se diseñó para mostrar Streaming y Audio en “Consumer Devices”, pero lo que sobre todo nos interesa: también permite la construcción de aplicaciones gráficas en Java a partir de un subconjunto de la AWT

## Introducción

- Havi usa algunos componentes de la AWT, evitando los llamados “pesados”, o que dependen de la plataforma en la que se ejecuta la aplicación, y añade un nuevo conjunto de widgets que basándose en aquellos realizan muchas tareas específicas relacionadas con gráficos que el receptor tendrá que efectuar.
- JavaSwing no es válido, al no formar parte de J2ME ni de pJava, es muy pesado.
- HAVI ofrece una implementación de Widgets ligeros partiendo de un subset básico de la AWT.
- Dos componentes principales: HComponent (hereda de java.awt.Component) y HContainer (de java.awt.Container)

## Introducción

- HAVI aporta además:
  - HLook: posibilidad de implementar el aspecto gráfico del componente
  - Matte: efectos gráficos avanzados. **Aunque dos cosas:** la primera es que es opcional (ya vimos en algún ejercicio que no se soportaba) y la segunda es que requiere de mucha CPU....
  - Gestión optimizada de Z-Ordering
- Todo lo anterior unido a la gestión de Transparencias nos ofrece una posibilidades gráficas importantes

## Widgets de HAVI

### INTERFACES

HActionable	HOrientable	HAnimateEffect	HLook
HNavigable	HComponentOrdering	HMatte	HAdjustableLook
HState	HTextLayoutManager	HMatteLayer	HExtendedLook
HSwitchable			

### UTILIDADES

HAnimateLook	HTextLook	HMultilineEntryLook	HFlatEffectMatte
HGraphicLook	HRangeLook	HFlatMatte	HImageEffectMatte
HListGroupLook	HSinglelineEntryLook	HImageMatte	

### COMPONENTES

HListGroup	HIcon	HMultilineEntry	HListElement
HRange(HStaticRange)	HGraphicButton	HRangeValue	HScene
HToggleGroup	HVideoComponent	HDefaultTextLayoutManager	HSinglelineEntry
HContainer	HSound	HAnimation (HStaticAnimation)	HStaticIcon
HVisible	HStaticText	HText	HTextButton
HComponent	HToggleButton		

## Soporte de Z-ordering potente

```
public interface HComponentOrdering{  
  
    public java.awt.Component addBefore(java.awt.Component component, java.awt.Component behind);  
    public java.awt.Component addAfter(java.awt.Component component, java.awt.Component front);  
    public boolean popToFront(java.awt.Component component);  
    public boolean pushToBack(java.awt.Component component);  
    public boolean pop(java.awt.Component component);  
    public boolean push(java.awt.Component component);  
    public boolean popInFrontOf(java.awt.Component move, java.awt.Component behind);  
    public boolean pushBehind(java.awt.Component move, java.awt.Component front);  
  
}
```

## Ciclo de Vida de la aplicación en el aspecto Gráfico HAVI

- Fuera del contexto de HAVI:
  - La plataforma lee la app y la valida: seguridad
  - Se instancia la JVM (en teoría una para cada app)
  - Se ejecuta
- Si no requiere un UI continua normal
- Si se requiere un UI, entonces se configuran los HScreenDevices (Graphics, Video, Background) con el pixel aspect ratio, la resolución deseada, etc.
- Se solicita acceso a parte de la pantalla mediante HSceneFactory, obteniendo un Container para la app: HScene
- La app usa la HScene para añadir sus componentes gráficos

## Ciclo de Vida de la Aplicación en el aspecto Gráfico HAVI

- La app puede saber si tiene el foco
- La app puede solicitar conocer si la HScene es ocultada, movida...
- La app puede re-establecer el tamaño de la HScene
- **La aplicación termina su presentación sobre la pantalla llamando a HScene.dispose(...)**
- Fuera del contexto de HAVi: La aplicación termina

## Ejercicios Bloque G2-1

## La fuente: Tiresias

- Interesante lectura: MHP 1.1.2 A0068r1: Appendix G.4 y D
- Esta fuente se definió como la más adecuada para personas con problemas de visión. Aunque hay quienes ponen esto en entredicho.
  - <http://screenfont.ca/fonts/today/Tiresias/>
- Por defecto: se proporcionará con peso PLAIN y Tamaño 26. Los nombres usados por la fuente soportada:
  - Nombre lógico:** "SansSerif" (devuelto por `java.awt.Toolkit.getFontList`).
  - Family Name:** "Tiresias" (devuelto por `java.awt.Font.getFamily`).
  - Font Face Name:** "Tiresias PLAIN".
- Java dice: en constructor de Font: **@param** name the font name. This can be a logical font name or a font face name. A logical name must be either: Dialog, DialogInput, Monospaced, Serif, SansSerif, or Symbol.
  - Forma de construirla** = `new Font( "Tiresias", Font.PLAIN, 24);`

## La fuente: Tiresias

- Ver en la tabla de abajo la relación entre puntos/pixels/recomendación de uso

Points (lo que se define en su creación) de 12 a 96	Altura en Pixels (tamaño de la V)	Recomendado para
36	24	Títulos / Subtítulos grandes
31	21	Subtítulos
26	18	Cuerpo
24	16	Pie de página

## Ejercicios Bloque G2-2

## HLook: cambiar el aspecto de nuestros componentes

- Los componentes HAVI ofrecen la posibilidad de establecerles **objetos “Pintores”** de los mismos. **La idea es evitar hacer herencia.**
- Básicamente lo que ofrece es por un lado una **serie de propiedades como MaximumSize, PreferredSize, Insets...** de forma que sean tenidas en cuenta por el objeto a la hora de pintarse, y por otro te da la posibilidad de **“pintar” en el Graphics antes de que él pinte el componente en sí**, pudiendo además evitar que lo haga.
- Cuando decimos que te da la posibilidad es que si lo deseas puedes hacerlo pero si no quieres, basta con que en el llamada al método en cuestión hagas un **super.método** y ya está.
- **Todos los HLook** existentes implementan al menos **la misma funcionalidad que el paint()**. O dicho de otra forma: Todos los componentes usan un HLook por defecto para pintarse.
- Otra importante posibilidad de HLook es que te notifica **cambios de propiedades del objeto** que puedan suponer un “repintado”.

## HLook. Operativa

- Se establece a nivel de **HVisible** (hereda de HComponent).
- **Ojo:** cada tipo acepta unos HLook determinados: no vale HGraphicLook para HText. Además **estos HLook** son los que “persé” ya usan los componentes para pintarse!! (ya comentado)
- Existe la posibilidad de establecer un **HLook por defecto por Clase:** método *static setDefaultHLook()*, de forma que no es necesario establecerlo para cada instancia de la clase. Siempre se puede hacer uno particularizado.

HText.setDefaultHLook( HTextLook ), **OJO:** el HLook por defecto establecido se usará por los componentes creados a partir de la asignación, **NO** hay efecto retroactivo! Se ve que se guardan el HLook la primera vez que se pintan.

- Existen propiedades de los objetos HAVI que sólo se tienen en cuenta por los objetos HLook, como por ejemplo **HTextButton.setBorderEnabled(...)**

## HLook. Operativa

- **Extensión:** Las clases de HLook existentes nos proporcionan mediante herencia la base sobre la cual podemos realizar nuestros propios HLook

## Ya hay HLooks implementados

### INTERFACES

HActionable	HOrientable	HTextLayoutManager	HLook
HNavigable	HAdjustmentValue	HMatte	HAdjustableLook
HState	HItemValue	HMatteLayer	HExtendedLook
HSwitchable	HTextValue	HComponentOrdering	HAnimateEffect

### UTILIDADES

HAnimateLook	HTextLook	HMultilineEntryLook	HFlatEffectMatte
HGraphicLook	HRangeLook	HFlatMatte	HImageEffectMatte
HListGroupLook	HSinglelineEntryLook	HImageMatte	

### COMPONENTES

HListGroup	HIcon	HMultilineEntry	HListElement
HRange(HStaticRange)	HGraphicButton	HRangeValue	HScene
HToggleGroup	HVideoComponent	HDefaultTextLayoutManager	HSinglelineEntry
HContainer	HSound	HAnimation (HStaticAnimation)	HStaticIcon
HVisible	HStaticText	HText	HTextButton
HComponent	HToggleButton		

**Clases a las que se pueden aplicar los Looks existentes**

HAnimation	<b>HAnimateLook</b>
HGraphicButton	<b>HGraphicLook</b>
HIcon	<b>HGraphicLook</b>
HListGroup	<b>HListGroupLook</b>
HMultilineEntry	<b>HMultilineEntryLook</b>
HOrientable	<b>HAdjustableLook</b>
HRange	<b>HRangeLook</b>
HRangeValue	<b>HRangeLook</b>
HSinglelineEntry	<b>HSinglelineEntryLook</b>
HStaticAnimation	<b>HAnimateLook</b>
HStaticIcon	<b>HGraphicLook</b>
HStaticRange	<b>HRangeLook</b>
HStaticText	<b>HTextLook</b>
HText	<b>HTextLook</b>
HTextButton	<b>HTextLook</b>
HToggleButton	<b>HGraphicLook</b>

## HLook: El API

- Nótese que en los métodos siempre se nos pasa el Componente que se está pintando en ese momento; **un HLook vale para muchos! Recordemos la asignación static del HLook.**
  - public void **showLook**(java.awt.Graphics g, HVisible visible, int state);
    - Redefiniendo el pintado.
    - Llamado desde el paint. (NO LLAMARLO DIRECTAMENTE!!)
    - Si no queréis hacer nada : super.showLook(...)
    - Si queréis: g.draw...y finalmente super.showLook(...) (si es que queréis)
  - public void **widgetChanged** (HVisible visible, HChangeData[] changes);
    - Llamado cuando su contenido, estado, o cualquier otro dato cambia.
    - Una implementación básica: **visible.repaint()**;
    - **Después veremos respecto a qué datos estamos hablando.**
  - public Dimension **getMinimumSize**(HVisible hvisible);
    - tamaño mínimo del componente cuando se usa este HLook

## HLook: El API

- public Dimension **getPreferredSize**(HVisible hvisible);
  - tamaño preferido del componente cuando se usa este HLook
- public Dimension **getMaximumSize**(HVisible hvisible);
  - tamaño máximo del componente cuando se usa este HLook
- public boolean **isOpaque**(HVisible visible);
  - Devuelve true si el área completa ocupada por este componente al pintarse con este HLook es completamente opaca
- public java.awt.Insets **getInsets**(HVisible visible)
  - Zona reservada por este HLook que rodeará al componente.
  - Se usará para pintar los bordes.

## Private HLook Data

- Este es un mecanismo de **implementación** para almacenar información, en teoría no debería de usarse. No obstante está ahí.
- El problema si lo usáis es que no sabemos cómo funcionará con él una implementación determinada de HAVi.
- En la Clase HVisible
  - public void **setLookData**(java.lang.Object key, java.lang.Object data)
    - **Key:** Clave que usará la clase HLook para acceder a la información.
    - **param:** El dato a almacenar
  - public java.lang.Object **getLookData**(java.lang.Object key)
    - Devuelve el valor

## Ejercicios Bloque G2-3

## HLook. Operativa

- Hemos visto que HLook nos ayuda a definir nuestra vista proporcionándonos la siguiente información:
  - En la llamada el método showLook (.....**int state**)
  - Evento **widgetChanged** (**HVisible visible**, **HChangeData[] changes**);
- Veamos qué es **HChangeData**...¿ De qué tipo de cambios me avisa ? Existen numerosos, como veremos a continuación, aunque es lo más habitual reaccionar con un `repaint()` al **HVisible**.

## Ejercicios Bloque G2-4

## HChangeData

- HChangeData: **contiene la información anterior al cambio**. Es una clase con los siguientes datos:
  - int **hint**: identificador del tipo de dato
  - Object **data**: el valor anterior al cambio del dato
- En HVisible están las constantes que nos dicen lo que puede contener dicho array. Cada tipo de objeto notificará lo que le corresponda.

**Como veréis nos notifican practicamente TODO!**

## HChangeData

ID	Object	Significado
STATE_CHANGE	Integer	oldState
CARET_POSITION_CHANGE	Integer	oldPosition
ECHO_CHAR_CHANGE	Character	oldEcho
EDIT_MODE_CHANGE	Boolean	oldEditMode
MIN_MAX_CHANGE	Integer[]	[oldMin, oldMax]
THUMB_OFFSETS_CHANGE	Integer[]	[oldMin, oldMax]
ORIENTATION_CHANGE	Integer	oldOrientation
ITEM_VALUE_CHANGE	Integer	oldValue
ADJUSTMENT_VALUE_CHANGE	Integer	oldIndex
LIST_CONTENT_CHANGE	HListElement[]	oldContent
LIST_ICONSIZING_CHANGE	Dimension	oldSize
LIST_LABELSIZING_CHANGE	Dimension	oldSize
LIST_MULTISELECTION_CHANGE	Boolean	oldSelection
LIST_SCROLLPOSITION_CHANGE	Integer	oldScrollPosition
SIZE_CHANGE	Integer	oldSize

## HChangeData: Más

ID	Object	Significado
BORDER_CHANGE	Boolean	oldBorderMode
REPEAT_COUNT_CHANGE	Integer	oldRepeatCount
ANIMATION_POSITION_CHANGE	Integer	oldValue
LIST_SELECTION_CHANGE	ListElement[]	oldSelectedElements
UNKNOWN_CHANGE	Integer	UNKNOWN_CHANGE
TEXT_CONTENT_CHANGE	Object[9]	[Integer changedState, String oldNORMAL_STATEtext, String oldFOCUSED_STATEtext, String oldACTIONED_STATEtext, String oldACTIONED_FOCUSED_STATEtext, String oldDISABLED_STATEtext, String oldDISABLED_FOCUSED_STATEtext, String oldDISABLED_ACTIONED_STATEtext, String oldDISABLED_ACTIONED_FOCUSED_STATEtext]

## HChangeData: y Más

ID	Object	Significado
GRAPHIC_CONTENT_CHANGE	Object[9]	[Integer changedState, Image oldNORMAL_STATEimage, Image oldFOCUSED_STATEimage, Image oldACTIONED_STATEimage, Image oldACTIONED_FOCUSED_STATEimage, Image oldDISABLED_STATEimage, Image oldDISABLED_FOCUSED_STATEimage, Image oldDISABLED_ACTIONED_STATEimage, Image oldDISABLED_ACTIONED_FOCUSED_STATEimage]
ANIMATE_CONTENT_CHANGE	Object[9]	[Integer changedState, Image[] oldNORMAL_STATEanimation, Image[] oldFOCUSED_STATEanimation, Image[] oldACTIONED_STATEanimation, Image[] oldACTIONED_FOCUSED_STATEanimation, Image[] oldDISABLED_STATEanimation, Image[] oldDISABLED_FOCUSED_STATEanimation, Image[] oldDISABLED_ACTIONED_STATEanimation, Image[] oldDISABLED_ACTIONED_FOCUSED_STATEanimation]
CONTENT_CHANGE	Object[9]	[Integer changedState, Object oldNORMAL_STATEcontent, Object oldFOCUSED_STATEcontent, Object oldACTIONED_STATEcontent, Object oldACTIONED_FOCUSED_STATEcontent, Object oldDISABLED_STATEcontent, Object oldDISABLED_FOCUSED_STATEcontent, Object oldDISABLED_ACTIONED_STATEcontent, Object oldDISABLED_ACTIONED_FOCUSED_STATEcontent]

## Estados....showLook (.....int state)

- Para facilitar la implementación, la consistencia, el aspecto y el funcionamiento de nuestras aplicaciones, HAVI ofrece un soporte de Estados (y eventos) para los componentes gráficos susceptibles de interactividad.
- Los estados posibles son **8**...¿ donde se definen?: **HState**

## HState. 8 Estados

- Referidos a
  - **Acciones:** se ha producido una acción sobre el componente
  - **Activado/Desactivado:** Activada o no la interacción con el usuario
  - **Foco:** tiene el foco o no.

**NORMAL\_STATE:** No existe Interacción con el componente.

**FOCUSED\_STATE:** Tiene el foco

**ACTIONED\_STATE:** Ha habido acción sobre el componente (click) pero NO tiene el foco

**ACTIONED\_FOCUSED\_STATE:** Ha habido acción y SI tiene el foco

**DISABLED\_STATE:** Está disabled. El usuario No puede interactuar con el componente

**DISABLED\_FOCUSED\_STATE :** Está disabled pero tiene el foco

**DISABLED\_ACTIONED\_STATE:** Ha habido acción y está disabled

**DISABLED\_ACTIONED\_FOCUSED\_STATE:** Ha habido acción, está disabled y tiene el foco

## Estableciendo información asociada al estado

- Se nos ayuda pudiendo establecer **determinado contenido** de los componentes asociado al estado en el que se encuentran.
- Muy útil para textos e imágenes (pushed...)
- En HVisible

```
public void setContent(Object object, int state) (contenido genérico libre)
```

```
public Object getContent(int state)
```

```
public String getTextContent(int state)
```

```
public void setTextContent(String string, int state)
```

```
public Image[] getAnimateContent(int state)
```

```
public void setAnimateContent(Image[] imageArray, int state)
```

## Estableciendo información asociada al estado

- En HVisible

```
public void setGraphicContent(Image image, int state)
```

```
public Image getGraphicContent(int state)
```

- Para establecer un valor para todos los estados se usa el “estado”

**HState.ALL\_STATES**

Este estado lo vamos a usar mucho!

- **Para saber en qué estado se encuentra (HVisible)**

```
public int getInteractionState()
```

**NO usad setInteractionState!!!! Es interno!!!**

## Ejemplos de Contenido y Estilos según el estado

Estado	Contenido si no aplica ese estado porque ese componente no lo admite	Estilo lógico a mostrar
NORMAL_STATE	Ninguno	Pintado normalmente
FOCUSED_STATE	Como NORMAL_STATE	Resaltado, bien con borde o con color distinto
ACTIONED_STATE	Como FOCUSED_STATE	Hundido
ACTIONED_FOCUSED_STATE	Como FOCUSED_STATE	Hundido y resaltado
DISABLED_STATE	Como NORMAL_STATE	Tonos grises
DISABLED_FOCUSED_STATE	Como DISABLED_STATE	Tonos grises pero resaltado
DISABLED_ACTIONED_STATE	Como ACTIONED_STATE	Tonos grises y hundido
DISABLED_ACTIONED_FOCUSED_STATE	Como DISABLED_STATE	Tonos grises, hundido y resaltado

## Gestionando el Foco

- Aquellos que implementen **org.havi.ui.HNavigable** podrán ser susceptibles de recibir el foco
- El hecho de recibir el Foco también implica que el usuario puede “navegar” de un componente que admite foco a otro. **Mediante este interface se puede establecer a qué componente “navegamos” cuando pulsan una determinada Tecla.**

## Clases que implementan HNavigable y generan Focus Events

HAnimation

HIcon

HText

HRange

HGraphicButton

HTextButton

HToggleButton

HListGroup

HSinglelineEntry

HMultilineEntry

HRangeValue

## HNavigable Interface

- Veamos el API
  - public interface **HNavigable** extends HNavigationInputPreferred{
    - HNavigationInputPreferred es de **implementación interna**.
  - public void **setMove**(int keyCode, HNavigable target)
    - Hacia qué objeto HNavegable se va si se pulsa el código pasado. **Ojo**: no confundir al usuario.
    - Sólo usad códigos que **org.havi.ui.event.HRcCapabilities.isSupported(key)==true**
    - **Conviene restringirse al set asegurado por defecto.**
  - public HNavigable **getMove**(int keyCode)
    - Hacia qué HNavegable se va si se pulsa ese código
  - public void **setFocusTraversal**(HNavigable up, HNavigable down, HNavigable left, HNavigable right)
    - Hacia donde para los 4 sentidos: VK\_UP, VK\_DOWN, VK\_LEFT, VK\_RIGHT
  - public boolean **isSelected**()
    - Si tiene el Foco

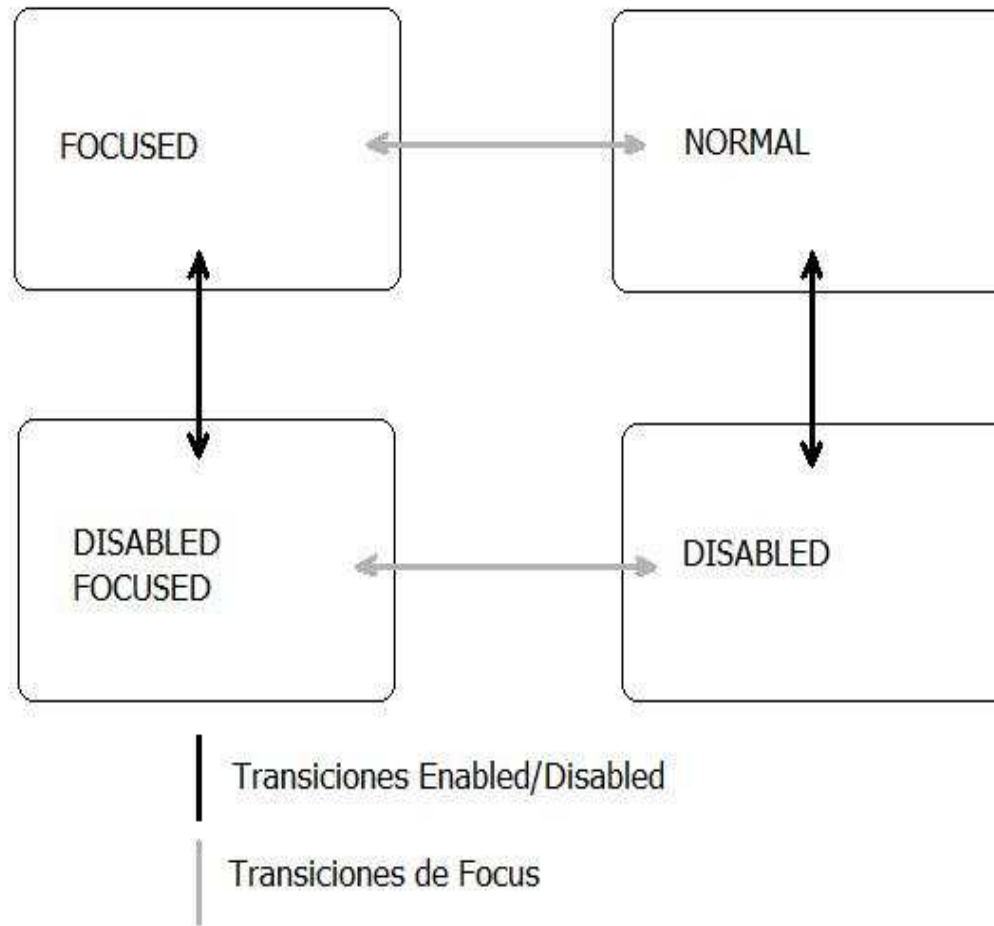
## HNavigable Interface y 2

- public void **setGainFocusSound**(HSound sound)
  - Establece el sonido a emitir cuando se adquiere el foco
- public void **setLoseFocusSound**(HSound sound)
  - Establece el sonido a emitir cuando se pierde el foco
- public HSound **getGainFocusSound**()
  - Sonido a emitir cuando se obtiene el foco
- public HSound **getLoseFocusSound**()
  - Sonido a emitir cuando se pierde el foco
- public void **addHFocusListener**(org.havi.ui.event.HFocusListener l)
- public void **removeHFocusListener**(org.havi.ui.event.HFocusListener l)
  - Gestión de Listener para eventos de foco

## HNavigable Interface y 3

```
public interface FocusListener extends EventListener {  
    public void focusGained(FocusEvent e);  
    public void focusLost(FocusEvent e);  
}
```

## Diagrama de estados de clases HNavigable



## Ejercicios Bloque G2-5

## Antes de entrar más en harina: Teclas soportadas por MHP

- Si tu app debe manejar alguna no listada a continuación debes establecer un mecanismo sustitutivo para el caso de que no se soporte. (¿ por qué complicarse ? Quizá si trabajamos sólo para un modelo concreto de STB -> ¿ solución propietaria ? )
- `org.havi.ui.event.HRcEvent` extends `java.awt.event.KeyEvent`

Tecla	Constante	Clase
Flecha arriba	VK_UP	java.awt.event.KeyEvent
Flecha abajo	VK_DOWN	java.awt.event.KeyEvent
Flecha derecha	VK_RIGTH	java.awt.event.KeyEvent
Flecha izquierda	VK_LEFT	java.awt.event.KeyEvent
Enter (= Select u OK)	VK_ENTER	java.awt.event.KeyEvent
Teclas de números	VK_0 A VK_9	java.awt.event.KeyEvent
Teletexto	VK_TELETEXT	org.havi.ui.event.HRcEvent
Primera coloreada	VK_COLORED_KEY_0	org.havi.ui.event.HRcEvent
Segunda Coloreada	VK_COLORED_KEY_1	org.havi.ui.event.HRcEvent
Tercera Coloreada	VK_COLORED_KEY_2	org.havi.ui.event.HRcEvent
Cuarta Coloreada	VK_COLORED_KEY_3	org.havi.ui.event.HRcEvent
<i>VK_COLORED_KEY_X se refiere a las hay en el mando de izquierda a derecha o de arriba abajo. Normalmente: Rojo, Verde, Amarillo Azul</i>		

¿ cómo sé que teclas soporta mi Deco ?

```
static boolean org.havi.ui.event.HRcCapabilities.isSupported(int keycode)
```

**También te ofrece un método para saber si hay un teclado virtual!**

```
static boolean org.havi.ui.event.HRcCapabilities.getInputDeviceSupported()
```

**Y dado que igual deseamos indicarle al usuario visualmente como es la tecla disponemos de una imagen o representación de la misma: Ver siguiente.**

## Representación de Teclas

```
static HEventRepresentation org.havi.ui.event.HRcCapabilities.getRepresentation(int aCode)
```

```
public class HEventRepresentation extends Object
```

```
    ER_TYPE_NOT_SUPPORTED = 0;
```

```
    ER_TYPE_STRING = 1;
```

```
    ER_TYPE_COLOR = 2;
```

```
    ER_TYPE_SYMBOL = 4;
```

```
    public boolean isSupported()
```

Si soporta el evento

```
    public int getType()
```

Tipo de representación que soporta: suma de las que soporta.

```
    public Color getColor()
```

Representación en color

```
    public String getString()
```

Representación en texto

```
    public Image getSymbol()
```

Representación en símbolo: **imagen de 32x32**

## Cuando usamos más teclas de las permitidas.....

- HAVI soporta muchísimas más teclas utilizadas para tareas normales que no tienen que ver con MHP.
- Es por esto que MHP exige que si una aplicación MHP usa teclas distintas a las soportadas, se le debe dejar claro al televidente que no se encuentra en lo que se llama “**TV Viewing Mode**” y que por lo tanto puede que las teclas no operen como espera.
- Cuando se da la circunstancia anterior la aplicación debe de ocupar al menos un 3% de la zona visible de la pantalla... “¿ será culpa del STB ? ¿ del TV Panasonic ? ¿ del emisor del canal ? ...Ah no! Va a ser la aplicación que aparece ahí debajo...”.

¿ cómo capturo eventos si no tengo el foco ?

**Gestión de eventos Reservados!! Lo veremos más adelante**

## **HActionable Interface. Ya hemos visto el Foco. Ahora Acciones!**

- Clases que implementan este Interface son las que pueden recibir acciones del usuario
- Como veremos toda clase **HActionable es HNavigable**: genera Focus Events además de los Action Events!:

```
public interface HActionable extends HNavigable
```

**Las Clases que implementan HActionable generan Action Events, y son:**

HGraphicButton

HTextButton

HToggleButton

## HActionable Interface

- Veamos el API
  - public interface HActionable **extends HNavigable**, HActionInputPreferred
    - HActionInputPreferred es de implementación interna
  - public void **addHActionListener**(org.havi.ui.event.HActionListener l);
  - public void **removeHActionListener**(org.havi.ui.event.HActionListener l);
    - Gestión de Listeners

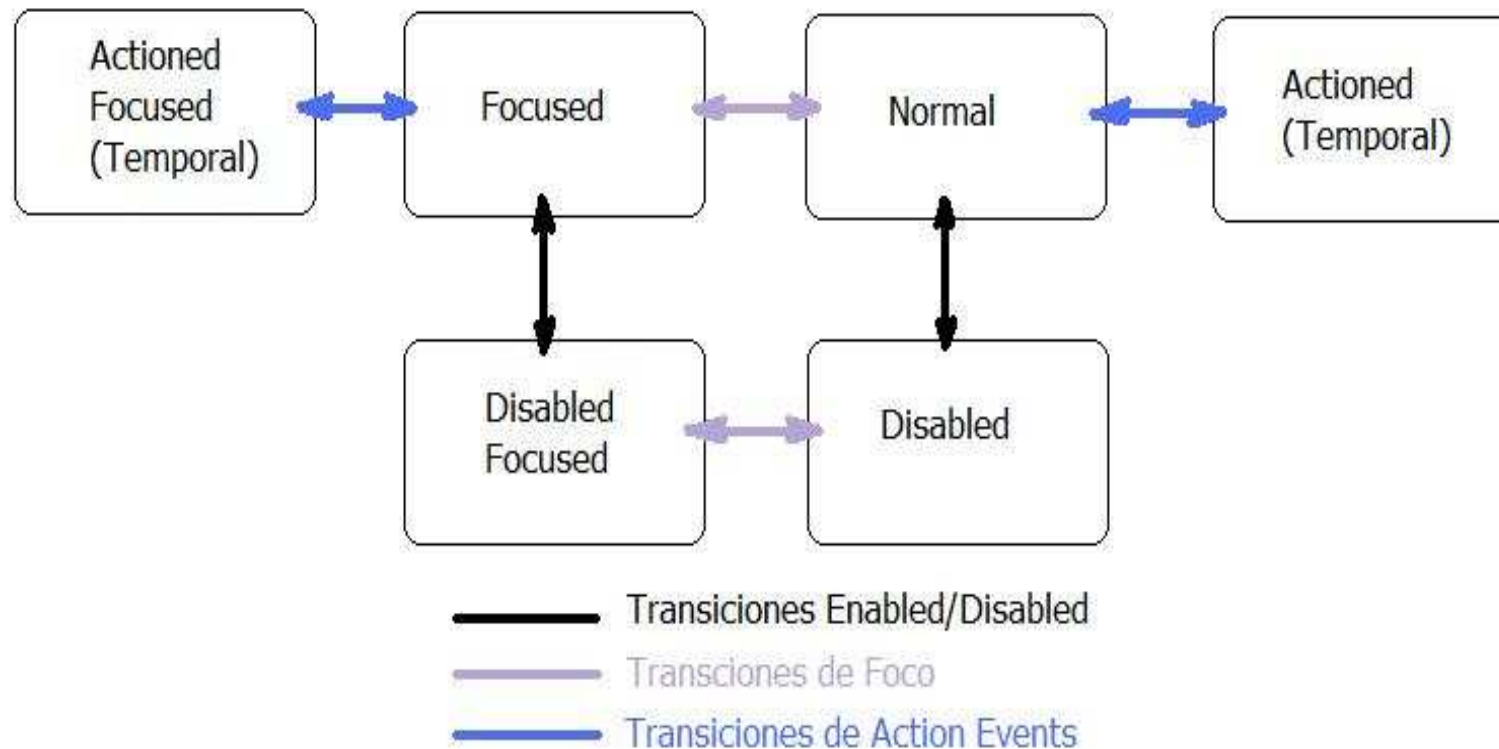
```
public interface HActionListener extends java.awt.event.ActionListener{}  
public interface ActionListener extends EventListener {  
    public void actionPerformed(ActionEvent e);  
}
```

- **java.awt.event.ActionEvent** ofrece información relevante respecto al Evento como el momento del mismo y si algunas teclas con ALT estaban pulsadas en el momento de producirse (no aplica mucho en nuestro contexto)

## HActionable Interface

- public void **setActionCommand**(String command);
  - Cuando se genera el evento HActionEvent recibido por el HActionListener se ofrece el método: `getActionCommand()` donde se devolverá este valor.
  - Más orientado a implementación.
- public void **setActionSound**(HSound sound);
- public HSound **getActionSound**()
  - Gestiona el sonido que se lanza en las transiciones:  
NORMAL\_STATE -> ACTIONED\_STATE,  
FOCUSED\_STATE -> ACTIONED\_FOCUSED\_STATE

## HActionable. Diagrama de Estados



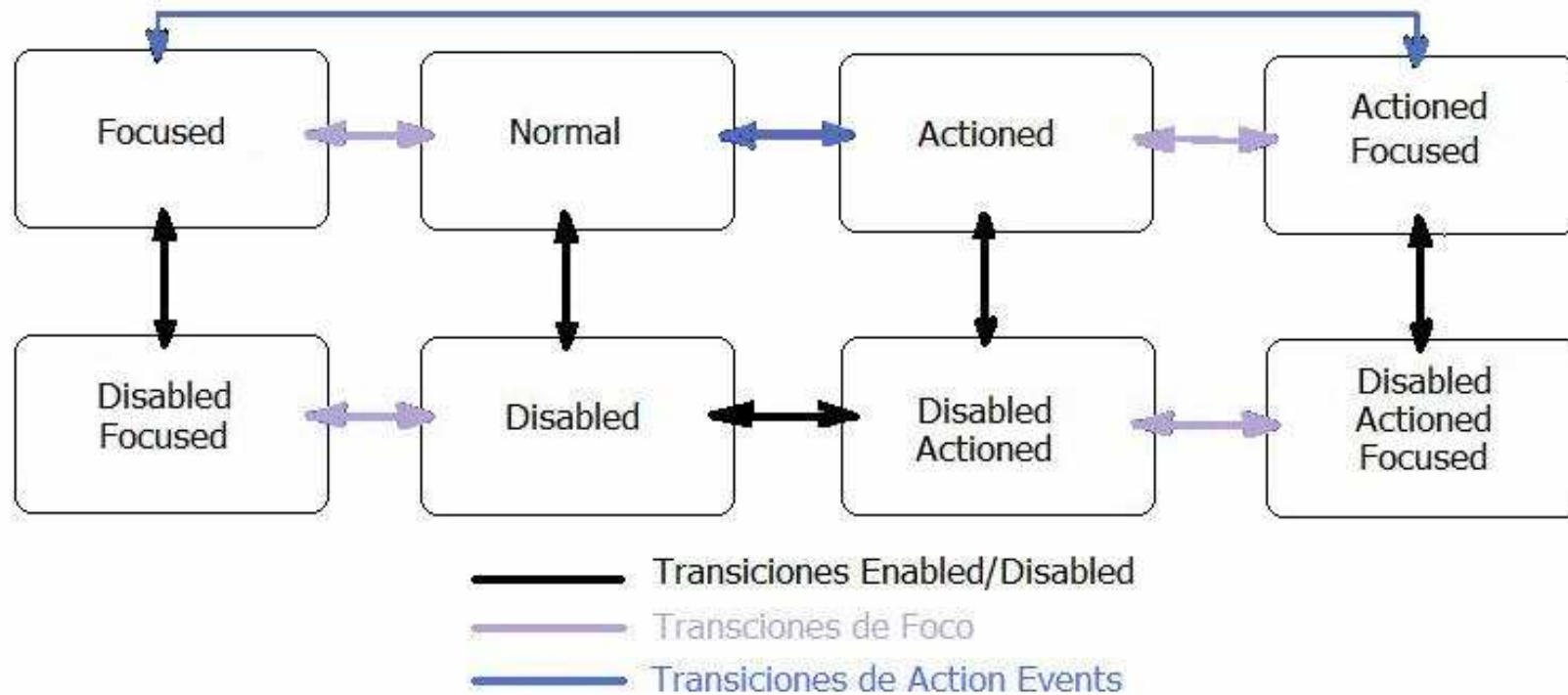
## Ejercicios Bloque G2-6

## HSwitchable Interface ¿ Cómo se gestiona un ToggleButton ?

- Este Interface lo implementan aquellas clases que admiten componentes de tipo Toggle, una (☺): **HToggleButton**
- Las Clases que implementan este Interface además gestionan **Foco y Acciones**
- Interface:

```
public interface HSwitchable extends HActionable {  
    public boolean getSwitchableState();  
    public void setSwitchableState(boolean state); Solo para Implementadores  
  
    public void setUnsetActionSound(HSound sound);  
    public HSound getUnsetActionSound();  
}
```

## Diagrama de Estados



## Ejercicios Bloque G2-7

### **HTextView, HAdjustmentValue, HOrientable, HItemValue**

- No establecen modelos de transición de estados, más bien definen una forma de trabajar con determinados componentes gráficos.
- Digamos que en lugar de trabajar directamente con una implementación lo hacemos contra un Interface, lo cual ofrece flexibilidad para la implementación en las anteriores.

## **HTextView: manejo de texto y eventos de teclado**

- Implementado por todos los componentes que pueden **editar Texto**. Te ofrece utilidades para detectar **eventos de teclado y de cambios en el texto**.
- Lo implementan **HSinglelineEntry** y **HMultilineEntry**
- El API

```
public interface HTextView extends HNavigable, HKeyboardInputPreferred{
```

```
    public void addHKeyListener(org.havi.ui.event.HKeyListener l);  
    public void removeHKeyListener(org.havi.ui.event.HKeyListener l);
```

- Gestión de Eventos de Teclado

```
    public void addHTextListener(org.havi.ui.event.HTextListener l);  
    public void removeHTextListener(org.havi.ui.event.HTextListener l);
```

- Gestión de Eventos de cambios en el Texto

```
}
```

```
public interface HTextListener extends java.util.EventListener{
```

```
    public abstract void textChanged(org.havi.ui.event.HTextEvent e);  
    public abstract void caretMoved(org.havi.ui.event.HTextEvent e);}
```

## HTextValue: manejo de texto y eventos de teclado

```
public interface HKeyListener extends java.awt.event.KeyListener{}  
public interface KeyListener extends EventListener {  
    public void keyTyped(KeyEvent e);  
    public void keyPressed(KeyEvent e);  
    public void keyReleased(KeyEvent e);  
}
```

- Las posibilidades de Eventos en cuanto a los cambios de Texto son importantes:  
ved [org.havi.ui.event.HTextEvent](http://org.havi.ui.event.HTextEvent)

```
TEXT_START_CHANGE  
TEXT_CHANGE  
TEXT_CARET_CHANGE  
TEXT_END_CHANGE  
CARET_NEXT_CHAR  
CARET_NEXT_LINE  
CARET_PREV_CHAR  
CARET_PREV_LINE  
CARET_NEXT_PAGE  
CARET_PREV_PAGE
```

## HTextView: manejo de texto y eventos de teclado

- **HKeyboardInputPreferred** permite establecer qué tipo de caracteres queremos que se admitan

```
public interface HKeyboardInputPreferred{  
    // Tipos de Caracteres admitidos  
    public static final int INPUT_NUMERIC = 1;  
    public static final int INPUT_ALPHA = 2;  
    public static final int INPUT_ANY = 4;  
    public static final int INPUT_CUSTOMIZED = 8;  
  
    public boolean getEditMode();  
    public void setEditMode(boolean edit);  
    // NO se debe llamar, sólo para implementadores. SI PERO MENOS, SI SE LLAMA: AYUDA  
    // BASTANTE AL EVITAR EL INTRO PARA ENTRAR EN UNA CAJA DE TEXTO  
  
    public int getType();  
    public char[] getValidInput();  
    // INTERNO. Proceso de Eventos.  
    public void processHTextEvent(org.havi.ui.event.HTextEvent evt);  
    public void processHKeyEvent(org.havi.ui.event.HKeyEvent evt);  
}
```

## Ejercicios Bloque G2-8

## HAdjustmentValue

- Aquellos que deben de ajustar de alguna manera un rango de valores numéricos.
- ¿ Quien implementa HAdjustmentValue ?

**HRangeValue**: el Slide Bar. Hereda de HRange y este a su vez de HStaticRange; ambos ofrecen un comportamiento base que no implementa HAdjustmentValue.

- El API:
  - public interface HAdjustmentValue extends **HNavigable**, HAdjustmentInputPreferred
    - Gestionan Foco: HNavigable.
  - public int **getUnitIncrement**();
  - public void **setUnitIncrement**(int increment);
    - Se establece el incremento; Si menos de 1 se considera 0.
    - En oo que debería de variar el valor cuando se reciban eventos de ADJUST\_LESS, ADJUST\_MORE

## HAdjustmentValue

- public void **setBlockIncrement**(int increment);
- public int **getBlockIncrement**();
  - Establece el incremento de salto de página; Si menos de 1 se considera 0.
  - En lo que debería de variar el valor cuando se reciban eventos de ADJUST\_PAGE\_LESS, ADJUST\_PAGE\_MORE
- public void **addAdjustmentListener**(org.havi.ui.event.HAdjustmentListener l);
- public void **removeAdjustmentListener**(org.havi.ui.event.HAdjustmentListener l);
  - Gestor de Listeners de Eventos.

```
public interface HAdjustmentListener{  
    public abstract void valueChanged(org.havi.ui.event.HAdjustmentEvent e);  
}
```

## HAdjustmentValue

- Los eventos que se gestionan son (ver org.havi.ui.event.HAdjustmentEvent ):
  - ADJUST\_START\_CHANGE
  - ADJUST\_LESS
  - ADJUST\_MORE
  - ADJUST\_PAGE\_LESS
  - ADJUST\_PAGE\_MORE
  - ADJUST\_END\_CHANGE
- public void **setAdjustmentSound**(HSound sound);
- public HSound **getAdjustmentSound**();
  - Sonido cuando cambia el valor

## **HAdjustmentValue**

- Así como HTextValue disponía de HKeyboardInputPreferred, HAdjustmentValue tiene **HAdjustmentInputPreferred (Ojo a HOrientable)**

```
public interface HAdjustmentInputPreferred extends HOrientable {  
    public boolean getAdjustMode(); // indica si está en modo de Ajuste o no  
    public void setAdjustMode(boolean adjust); // Para implementadores!!!!  
    public void processHAdjustmentEvent(org.havi.ui.event.HAdjustmentEvent evt); // Interno  
}
```

- **HOrientable:** Implementado por componentes que soportan Orientación

```
public interface HOrientable{  
    ORIENT_LEFT_TO_RIGHT  
    ORIENT_RIGHT_TO_LEFT  
    ORIENT_TOP_TO_BOTTOM  
    ORIENT_BOTTOM_TO_TOP  
    public int getOrientation();  
    public void setOrientation(int orient);  
}
```

## Ejercicios Bloque G2-9

**¿ Cómo capturo un evento de teclado si no tengo el Foco ?**

- Para evitar problemas de foco (no quitárselo a otros Xlets) las HScenes se crean invisibles
- Necesitamos poder capturar eventos de teclado aún sin tener el foco, es más, incluso sin existir componentes gráficos creados... ¿ cómo ?  
Implementando el Interface: **org.dvb.event.UserEventListener**
- Por el momento sólo se gestionan eventos de teclado
- En general este modelo de suscripción es notablemente más fiable que la suscripción directa a eventos de un componente en concreto (addKeyListener).

## Proceso

- **Paso 1:** se crea objeto del tipo: [org.dvb.event.UserEventRepository](#)
- **Paso 2:** se indican los códigos de teclas respecto a los que queremos recibir eventos.

Existen varias opciones:

```
public void addAllArrowKeys()  
public void addAllColourKeys()  
public void addAllNumericKeys()  
public void addKey(int keycode)  
public void removeAllArrowKeys()  
public void removeAllColourKeys()  
public void removeAllNumericKeys()  
public void removeKey(int keycode)
```

## Proceso

- **Paso 3:** nos suscribimos a los eventos de la siguiente forma:
  - `org.dvb.event.EventManager.getInstance().addUserEventListener((UserEventListener)nuestroListener, (UserEventRepository)nuestroRepository)`
  - >Es un singleton
  - >Hemos de implementar el Listener
  - **Para de-suscribirnos:** `public void removeUserEventListener(UserEventListener listener)`

## UserEventListener

```
public interface UserEventListener extends java.util.EventListener {  
    public void userEventReceived (UserEvent e) ;  
}
```

- Los más relevante de UserEvent:

- public int getType ()
  - **KEY\_PRESSED, KEY\_RELEASED or KEY\_TYPED.**
    - **OJO cuando gestionéis la recepción de eventos!!! además del Code hay que controlar el type pues llega más de un evento para cada pulsación.**
- public int getCode ()
  - El código del evento, p.e. VK\_DOWN. Para eventos de teclado tipo KEY\_TYPED (combinación de pressed y released que generan chars) es VK\_UNDEFINED
- public char getKeyChar()
  - Carácter que representa la tecla
- public int getModifiers()
  - si los hay.
- public boolean isShiftDown()
- public boolean isControlDown()
- public boolean isMetaDown()
- public boolean isAltDown()
- public long getWhen()

## Ejercicios Bloque G2-10

## Acceso Exclusivo

- Podría ocurrir que necesitásemos asegurarnos de que ninguna otra aplicación va a recibir eventos de teclado de nuestra app, por ejemplo si estamos introduciendo una password, número de cuenta, etc....
- Podemos hacer que SOLO nuestra app reciba eventos determinados de teclado de dos formas mediante el singleton **org.dvb.event.EventManager**

### A) Suscribimos a los eventos usando el método que incluye un **ResourceClient**:

```
public boolean addEventListener (UserEventListener listener,  
    org.davic.resources.ResourceClient client,  
    UserEventRepository userEvents)
```

- Cuando nos de-suscribimos se libera el acceso restringido:

```
public void removeEventListener(UserEventListener listener)
```

## Acceso Exclusivo

**B)** o si queremos usar la gestión de eventos normales de la AWT, llamando al método de `org.dvb.event.EventManager`:

```
public boolean addExclusiveAccessToAWTEvent  
(org.davic.resources.ResourceClient client, UserEventRepository userEvents)
```

– Cuando nos de-suscribimos se libera el acceso restringido:`public void  
removeUserEventListener(UserEventListener listener)`

```
public void removeExclusiveAccessToAWTEvent (org.davic.resources.ResourceClient client)
```

## **HItemValue**

- Aquellos que gestionan listas
- ¿ quién implementa el Interface ?

### **HListGroup**

- Veamos el API
  - public interface **HItemValue** extends **HNavigable**, **HSelectionInputPreferred** {
    - Soporta Foco
  - public void **addItemListener**(org.havi.ui.event.HItemListener l);
  - public void **removeItemListener**(org.havi.ui.event.HItemListener l);
    - Gestión de Listeners
  - public void **setSelectionSound**(HSound sound);
  - public HSound **getSelectionSound**();
    - Sonido para cuando se seleccione

## HItemValue

- Veamos el API

```
public interface HItemListener extends java.util.EventListener {  
    public void selectionChanged(org.havi.ui.event.HItemEvent e);  
    public void currentItemChanged(org.havi.ui.event.HItemEvent e);  
}
```

- El tipo de eventos que se pueden recibir es importante: Ved **org.havi.ui.event.HItemEvent** (los ofrece el método getType)
  - ITEM\_START\_CHANGE
  - ITEM\_TOGGLE\_SELECTED
  - ITEM\_SELECTED
  - ITEM\_CLEARED
  - ITEM\_SELECTION\_CLEARED
  - ITEM\_SET\_CURRENT
  - ITEM\_SET\_PREVIOUS
  - ITEM\_SET\_NEXT
  - SCROLL\_MORE
  - SCROLL\_LESS
  - SCROLL\_PAGE\_MORE
  - SCROLL\_PAGE\_LESS
  - ITEM\_END\_CHANGE

## **HItemValue**

- Veamos el API
  - Interfaz para implementers. No así **HOrientable**

```
public interface HSelectionInputPreferred extends HOrientable{  
    public boolean getSelectionMode();  
    public void setSelectionMode(boolean edit); // no usar  
    public void processHItemEvent(org.havi.ui.event.HItemEvent evt); // no usar.  
}
```

## Ejercicios Bloque G2-11

## Perfil principal de los componentes

Componente	Interface
HAnimation	<a href="#">HNavigable</a>
HGraphicButton	<b>HActionable</b>
HIcon	<a href="#">HNavigable</a>
HListGroup	HItemValue
HMultilineEntry	<a href="#">HTextValue</a>
HRange	<a href="#">HNavigable</a>
HRangeValue	HAdjustmentValue
HSinglelineEntry	<a href="#">HTextValue</a>
HStaticAnimation	HVisible
HStaticIcon	HVisible
HStaticRange	HVisible
HStaticText	HVisible
HText	<a href="#">HNavigable</a>
HTextButton	<b>HActionable</b>
HToggleButton	HSwitchable

## **Transparencia. Muy útil en el contexto de la TV**

- Es muy útil que HScene sea transparente por defecto. Ya hemos visto cómo se pueden crear imágenes con ciertos niveles de transparencia.
- ¿ Cómo hacer que un determinado componente tenga una determinada transparencia que nos permita ver **a través de la capa de Graphics y de Video ?**

- Si por ejemplo creamos una imagen con un Color con una determinada transparencia y la pintamos, esta se pintará con la transparencia indicada y veremos lo que haya detrás:
  - Si la HScene no tiene nada configurado detrás: el VideoDevice
  - Si la HScene tiene un Color de BackGround este se ve, aunque esté en mode NO\_BACKGROUND\_FILL. Es Curioso que el Color de background se ve si pintamos una imagen, y sin embargo en el resto del HScene no.
  - Si la HScene tiene una Imagen pues veremos esta. Y si esta es a su vez transparente veremos la mezcla de ambas y de fondo el video.

## Ejercicios Bloque G2-12

- **Atención:** ni el método siguiente ni sus variantes se deben usar, de hecho la implementación no lo soporta: **MHP 1.1.2 A0068r1, 11.4.1.2**
  - The methods `HGraphicsConfiguration.getPunchThroughToBackgroundColor(int percentage)` shall **not** be used by inter-operable applications.
  - Este método te devuelve un `Color` que puedes usar para pintar sobre `Graphics` y lo que hace es que aplica el porcentaje de Transparencia pedida (0-100) sobre los pixels sobre los que pintas, hasta el `Background`.
- Usar el modelo de Colores transparentes de manera individual como acabamos de ver, que por otro lado no se soporta! (☺), puede resultar muy complicado y arriesgado, pero tenemos dos opciones más: **Mattes** y **DVBGraphics**
  - `DBVGraphics`: permite establecer el comportamiento de un objeto con respecto a los ya pintados en cuanto a transparencias se refiere.
  - `Mattes`: viene a ofrecer una capa de pintado de `background` de los componentes con propiedades de transparencias

## DVBAAlphaComposite

- Se puede establecer un esquema de transparencias usando **org.dvb.ui.DVBAAlphaComposite**

```
DVBAAlphaComposite dva = DVBAAlphaComposite.getInstance(DVBAAlphaComposite.SRC_IN,0.5f);
```

```
((DVBGraphics)g).setDVBComposite(dva);
```

- Mediante DVBAAlphaComposite establecemos cómo queremos que se comporte el Graphics del componente que vamos a pintar **en relación con los ya pintados**. Veamos los dos parámetros:
  - **Porter-Duff Rules**, es el **modelo de representación** (lo vemos en siguiente Slide): Este es el primer parámetro de  

```
DVBAAlphaComposite.getInstance(DVBAAlphaComposite.SRC_IN,0.5f);
```
  - El segundo parámetro es el Alpha (float de 0-1) que nos define el **nivel de transparencia** (1 = opaco)  

```
DVBAAlphaComposite.getInstance(DVBAAlphaComposite.SRC_IN,0.5f);
```

## DVBAAlphaComposite

- El nivel de Alpha de lo que finalmente se pinta se calcula a partir de 3 valores:
  - Alpha del pixel fuente
  - Alpha del pixel destino
  - Alpha de DVBAAlphaComposite establecido en el Graphics cuando se pinta la fuente sobre el destino

## Tipos de Porter-Duff Rules soportados en MHP

rule	Efecto
SRC	La fuente se pinta sobre el destino. Cualquier área solapada se sustituye por la fuente
CLEAR	Color y áreas solapadas de la fuente se eliminan. El source no solapado con el destino desaparece
SRC_IN	El Color del área coincidente se toma de la fuente. Las áreas que no están solapadas de la fuente no se pintan
SRC_OUT	El Color del área coincidente se toma de la fuente.
DST_IN	El Color del área solapa se toma del destino. Las áreas de la fuente que no estén solapadas no se pintan.
DST_OUT	Similar a DST_IN pero los cálculos para el Alpha resultante son diferentes
SRC_OVER	La fuente se superpone sobre el destino.
DST_OVER	El destino se superpone sobre la fuente.

## Ejemplos

- El rectángulo traslúcido se ha pintado sobre una imagen en memoria usando SRC.
- A su vez el círculo se ha pintado sobre esta imagen en memoria usando los diferentes rules. Una vez que se ha pintado el círculo, la imagen se pinta usando SRC\_OVER



- Trabajando con Componentes que directamente pintan sobre el Graphics mediante herencia de HComponent / HContainer sí se han obtenido resultados, pero ni mediante herencia de por ejemplo HText o mediante HLook ha sido posible que un componente se pintase con un DVBAAlphaComposite.

## Mattes

- Mattes no tienen porqué soportarse, de hecho, en el STB donde se ejecuta el XLET no se soportan. No se va a entrar en detalle sobre ellos.

### Anex G.7

- HAVi mattes: Platforms are not required to implement the functionality of mattes in HAVi. Non implementation should be implemented as specified by HAVi.

org.havi.ui.HMatteException: mattes not supported

```
[2#1:2]      at org.havi.ui.HComponent.setMatte(Unknown Source)
[2#1:2]      at code4tv.mhp112.exercise20.Exercise20.configureContenedor(Unknown Source)
[2#1:2]      at code4tv.mhp112.exercise20.Exercise20.setComponents(Unknown Source)
[2#1:2]      at code4tv.mhp112.exercise20.Exercise20.setHScene(Unknown Source)
[2#1:2]      at code4tv.mhp112.exercise20.Exercise20.run(Unknown Source)
[2#1:2]      at java.lang.Thread.startup(Unknown Source)
```

Cuando se describen las opciones de HGraphicsDevice:

```
MATTE_SUPPORT=5 pref obj =null (REQUIRED_NOT)
```

## Mattes

- Existen varios:
  - HFlatEffectMatte
  - HFlatMatte
  - HImageEffectMatte
  - HImageMatte
- Echad un ojo a los APIs.

## Ejercicios Bloque G2-13

- Debido a las variaciones de aspect ratio de screen y de pixels la representación del texto puede variar de una configuración a otras (parcialmente resuelto con 14:9)
- La correspondencia de un punto de FontMetrics con respecto a los pixels es de 1:1 en vertical pero variable en horizontal. Esto ofrece problemas por ejemplo para saber cuando hace falta wrapping o no, o en qué medida se produce.
- Con el fin de facilitar el trabajo de presentación de texto MHP ofrece la clase **org.dvb.ui.DVBTextLayoutManager** la cual soporta muchísimas posibilidades (Ver Annex D 1.1.2 A0068r1)

- Las funcionalidades básicas se encuentran en el interface `org.havi.ui.HTextLayoutManager`, el cual implementa `org.havi.ui.HDefaultTextLayoutManager` y que es la que tienen por defecto componentes como `HText`, `HTextButton` o `HStaticText`

```
public interface HTextLayoutManager{  
    public void render(String markedUpString, java.awt.Graphics g, HVisible v, java.awt.Insets insets);  
}  
  
public void setTextLayoutManager(HTextLayoutManager manager)
```

- Este **HDefaultTextLayoutManager** ofrece unas posibilidades de Layout muy básicas:
  - Gestión de tamaño máximo, mínimo y preferido del componente y
  - se comporta de la siguiente forma cuando el texto “no cabe”: presenta “(…)” para indicar al usuario que no está viendo todo el texto
- El establecimiento del `HTextLayoutManager` se encuentra a nivel de `HVisible`. Basta con instanciar `HDefault...` y asignarlo al componente.

```
public void setTextLayoutManager(HTextLayoutManager manager)
```

- Si queremos buenas posibilidades de Rendering de Texto hemos de usar: **org.dvb.ui.DVBTextLayoutManager**. Ofrece las siguientes funcionalidades

## Alineación horizontal del texto

```
public static final int HORIZONTAL_START_ALIGN: izda
public static final int HORIZONTAL_END_ALIGN: dcha
public static final int HORIZONTAL_CENTER: centrado hori.
```

## Alineación vertical del texto

```
public static final int VERTICAL_START_ALIGN: arriba
public static final int VERTICAL_END_ALIGN: abajo
public static final int VERTICAL_CENTER : centrado vert.
```

## Orientación de la línea

```
public static final int LINE_ORIENTATION_HORIZONTAL: horiz.
public static final int LINE_ORIENTATION_VERTICAL : vertical
```

## Donde se comienza a escribir

```
public static final int START_CORNER_UPPER_LEFT: esq. superior izda.
public static final int START_CORNER_UPPER_RIGHT: esq. superior dcha.
public static final int START_CORNER_LOWER_LEFT: : esq. inferior izda.
public static final int START_CORNER_LOWER_RIGHT: esq. inferior dcha.
```

## Espacio entre líneas

```
setLineSpace: -1 es lo que dicte la fuente
```

## Espacio entre caracteres

```
setLetterSpace: unidades de 1/256 el espacio entre chars
```

## Text Wrapping S/N

```
setTextWrapping
```

## Espacio de tabulación

```
setHorizontalTabSpacing
```

## Insets: márgenes

```
setInsets
```

## Notifica cuando el texto no cabe

```
addTextOverflowListener
```

## Ejercicios Bloque G2-14

- Al margen de Tiresias, que es la fuente que se soporta por defecto, es posible cargar dinámicamente otras fuentes que se adapten a caracteres de otras zonas: son las de tipo PFR, Portable Font Resource.
- La carga de fuentes se realiza creando una factory del tipo `org.dvb.ui.FontFactory` para cada proveedor de Fonts (URL). Ver JavaDoc y A0068r1, 7.4,

## **Imágenes. Los tipos que se soportan son**

**GIF: NO es requerido**

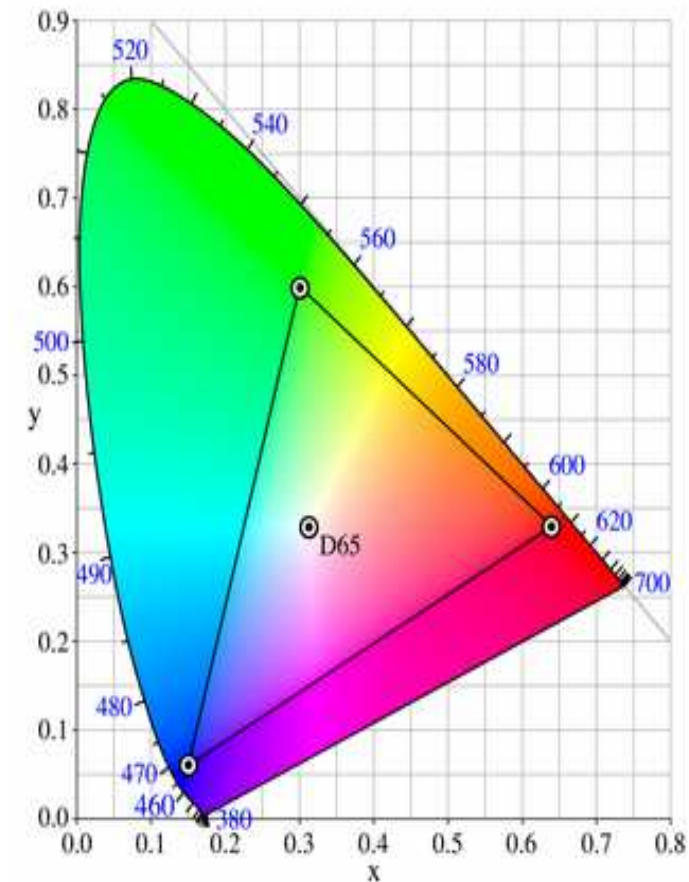
PNG: con restricciones. Ved MHP 1.1.2, 15.1

JPG

MPEG-I Frames

- El espacio de color que mejor representa los colores en la TV, Internet y en Monitores es **sRGB**.
  - IEC 61966-2-1: "Multimedia systems and equipment - Colour measurement and management - Part 2-1: Colour management – Default RGB colour space - sRGB".
- Un espacio de colores está definido por un rango concreto de colores del mismo.
- El espacio de colores, el Gamut, en el se codificarán los colores en el entorno MHP será sRGB.
- Así pues, mientras que se asuma que los colores no se salen de este gamut, es posible realizar las transformaciones que sean oportunas dentro de este rango.
- Anex G. 1.5: "The colour model is a "true colour" one. There shall be at least 4 bits per pixel for each of R. G and B."

[http://en.wikipedia.org/wiki/Image:CIExy1931\\_sRGB.svg](http://en.wikipedia.org/wiki/Image:CIExy1931_sRGB.svg)



- Todos los Textos en MHP están en UTF-8 de forma que la presentación de caracteres especiales es “Straight-forward”

- Intentaremos trabajar en el capítulo referido a JMF con HVideoComponent: componente aparte que se usa para manipular fácilmente la posición y tamaño de una emisión de vídeo colocándola en una jerarquía AWT,
- No obstante: MHP 1.1.2, A068r1
- **NO debemos asumir que será el componente devuelto por un JMF Player. Incluso nos recomiendan no usarlo si queremos tener una app Interoperable**
  - A.7.4.32 HVideoComponent
    - A.7.4.32.1 HAVi Specification
      - In section 8.3.3.6 "Integrating HAVi Video Support into Platforms" the following two sentences shall be removed:
        - "The class HVideoComponent is intended to be returned by a platform specific controller for video. In platforms based on the Java Media Framework, the Player.getVisualComponent method shall return objects of this class."
      - Note that the HVideoComponent class must be present, and MHP terminals may choose to use it or not. **Interoperable applications should not use HVideoComponent.**

<b>ISO/IEC 13818-1</b>	Part 1. Elementary Streams transport definition
<b>ISO/IEC 13818-6</b>	Part 6. Extensions for DSM-CC. Digital Storage Media Command and Control
<b>ETSI EN 300 468</b>	Digital Video Broadcasting (DVB);Specification for Service Information (SI) in DVB systems
<b>ETSI EN 301 192</b>	DVB specification for data broadcasting
<b>ETSI TR 101 202</b>	Implementation Guidelines for Data broadcasting
<b>ETSI TR 101 162</b>	Digital broadcasting systems for television, sound and data services; Allocation of Service Information (SI) codes for Digital Video Broadcasting (DVB) systems
<b>ETSI TR 102 154</b>	Implementation guidelines for the use of MPEG-2 Systems, Video and Audio in Contribution and Primary Dist
<b>ETSI TR 101 211</b>	Guidelines on implementation and usage of Service Information (SI)
<b>ETSI TR 101 200</b>	Digital Video Broadcasting (DVB); A guideline for the use of DVB specifications and standards
<b>DAVIC</b>	Digital Audio Visual Council. davic 1.4.1
<b>HAVI</b>	Specification of the Home Audio/Video Interoperability (HAVi) Architecture
<b>Interactivetvweb</b>	<a href="http://www.interactivetvweb.org/">http://www.interactivetvweb.org/</a>
<b>Wikipedia DSMCC</b>	<a href="http://en.wikipedia.org/wiki/DSM-CC">http://en.wikipedia.org/wiki/DSM-CC</a>
<b>MHP 1.1.2</b>	Multimedia Home Platform, A068r1 & tam668r23_11xdraft_20061115
<b>MHP 1.1.3</b>	Multimedia Home Platform, A068r3
<b>CDC 1.1</b>	Connected Device Configuration (CDC) 1.1 (JSR=218).
<b>PBP 1.1</b>	Personal Basis Profile 1.1 (JSR 217)
<b>MHP.org</b>	<a href="http://www.mhp.org">www.mhp.org</a>
<b>INTRO MHP 1.1.3</b>	tam1032r1-mhp-iptv-presentation