MoviePlayAir MANUAL

Requisitos Android

- Aplicación "**Adobe AIR**", se encuentra en el Market, es gratis.
- Aplicación "**ES File Manager**" para navegar en las carpetas compartidas de otros ordenadores en la red local. Se encuentra en el Market, es gratis.
- Se recomienda tener a mano el **Task Manager** (p.e. enlace directo en la home) para terminar la aplicación después de salir o cuando sea necesario reiniciarla. Al salir normalmente queda abierta en background. Creo que TarkManager ya viene instalado por defecto.

1 Instalación

La aplicación se llama **MoviePlayAIR** y se instala simplemente abriendo el archivo **MoviePlayAIR.apk** desde My Files o el ES File Manager. Por ejemplo, navegar con ES File Manager a una carpeta compartida de otro ordenador en el que se haya previamente guardado MoviePlayAIR.apk.

Puedes descargarla en: Systorgy.hangar.org

2 Ejecución

Sencillamente lanzar la aplicación MoviePlayAIR desde Aplicaciones o desde el enlace directo.

Recordemos que (al menos hasta que se implemente el protocolo de negociación para la conexión con Pol), para que MoviePlayAIR pueda conectarse con POL tenemos que **editar a mano el archivo movieplayer.cfg** y ponerle la IP de Pol Control.

3 Archivos y carpetas

A parte de la aplicación en sí, todos los archivos necesarios para el funcionamiento y la customización del espectáculo están en la **carpeta "pol" dentro de Documentos**. La carpeta de documentos en el Galaxy Tab se llama /sdcard/, no sé si se llama igual en todos los Android, pero es la que se abre por defecto cuando vas a "My Files". En My Files se muestra como "/Root", en ES File Manager como /sdcard/. Si no existe la carpeta "pol", la aplicación la creará automáticamente.

La aplicación utiliza/necesita los siguientes archivos (todos en la carpeta "pol"):

- **interface.swf:** es el archivo flash con el diseño customizable de la interfaz (botones, sliders, etc). No contiene programación (código): es la aplicación quien "escanea" el swf y le añade los comportamientos estándar. Reemplazar este archivo para cambiar la apariencia de la interfaz, añadir o quitar controles y por lo tanto variables.
- **movieplayer.cfg:** archivo de texto con la información para conectarse a POL. Es el archvo que normalmente generaría POL y depositaría (a través de la red) en la carpeta del player. Sin embargo con Android esto no se puede hacer fácilmente. De momento hay que editar este archivo **a mano** (se puede hacer desde el propio Android) y poner la IP del Pol Control. Este archivo quedará obsoleto en cuanto tengamos impelmentado el protocolo de negociación via UDP con Pol Control.
- **(opcional) un archivo .swf o .jpg por cada escena:** los nombres de los archivos deben ser los que se ponen en Pol Setup en la configuración de escena. Si son archivos swf, pueden tener controles (botones/sliders...) para variables específicas de cada escena (no se necesita programación para ello, ver capítulos siguientes) y pueden tener programación para comportamientos arbitrarios que se salgan del estándar. Si se usan jpg pueden servir de imagen de fondo de pantalla, como

manera más *guay* de saber en qué escena estás.

Cuando instalamos la aplicación, esta lleva consigo (dentro de la carpeta en la que se instala, que es inaccesible en Android) una **versión por defecto de interface.swf y movieplayer.cfg**. Cuando se ejecuta la aplicación, si no existen estos archivos en la carpeta "pol", la aplicación copiará automáticamente las versiones de fábrica a la carpeta pol. Así que no os extrañéis si borráis estos archivos y vuelven a aparecer mágicamente.

4 Principios de funcionamiento

Desde el punto de vista de POL, el controlador Android es **un Flash Player más**. Su función es enviar y recibir las que solemos llamar "variables" según nuestra nomenclatura obsoleta (más correctamente serían *mensajes*).

El uso más inmediato que se le puede dar es el de un **mini-joydreske:** un dispositivo con un cierto número de variables preestablecidas y que sirven para todo el espectáculo y para cualquier espectáculo (aunque en POL se le asignen comportamientos distintos en cada escena): M botones on/off y N ejes con rangos estándar (0-100), de la misma manera que el joydreske tiene 2 ejes y 10 botones y siempre son esos. Estos controles estándar incluyen el **acelerómetro** y el **micrófono** que corresponden a unas variables con nombres preestablecidos, de rango y on/off (ver capítulo sobre micrófono y acelerómetro).

Sin embargo, el tipo y número de variables y el aspecto gráfico de la interfaz se pueden fácilmente **customizar** para cada espectáculo e incluso **para cada escena**. La customización se hace **editando archivos flash** (fla) y reemplazando los archivos swf en la carpeta "pol". No es necesario programar en ActionScript para customizar la interfaz (aunque se puede usar programación para añadir comportamientos más complejos o específicos fuera del estándar).

Por lo tanto el controlador puede llegar a ser una interfaz completamente adaptada al espectáculo y que cambia su apariencia y funcionalidad en cada escena, ofreciendo siempre sólo los controles relevantes en cada momento (entre otras ventajas).

La comunicación entre el dispositivo y POL es totalmente bidireccional y por lo tanto el dispositivo sirve también para monitorizar lo que está pasando.

Existen nombres convencionales establecidos para mensajes especiales de POL a Android que sirven para dar órdenes al Android cuales por ejemplo: hacer visible/invisible el botón/control de una variable; activar/desactivar el envío de las variables de acelerómetro y micrófono; cambiar el modo de funcionamiento de un botón (push/toggle), cambiar el umbral de detección del micrófono o del acelerómetro, etc. Estos mensajes se llaman **variables especiales**.

4.1 Cambio de escena

En la comunicación **de POL a Flash** el cambio de escena es un evento especial, no es equivalente a una variable cualquiera. Sin necesidad de añadir eventos automáticos "begin", POL siempre informa a todos los Flash Players (y el Android es uno de ellos) de cada cambio de escena; o según se quiera ver, les *ordena* que cambien de escena. En POL Setup cada escena lleva asociado, por cada Flash Player, un **nombre de archivo (supuestamente) swf.** Siguiendo la tradición, el MoviePlayAIR para Android, cuando recibe un cambio de escena, intenta cargar el archivo con ese nombre. Si es un archivo swf, puede contener botones y controles para variables específicas de la escena. Este archivo se cargará "dentro" de la inerfaz principal, en un contenedor preparado y posicionado para ello: así dependiendo de cómo se haya diseñado la interfaz, los controles de la escna pueden fundirse con los controles globales (delante o detrás de ellos), o estar en una "página"

a parte. También se puede usar el archivo de escena como simple fondo de pantalla: en este caso puede ser tanto un swf como una imagen (jpg o png).

Otro uso que se le puede dar al nombre de archivo asociado a la escena (en POL Setup) es usrlo como **nombre de escena**: si no necesitamos controles customizados escena por escena sino que tenemos unos controles estándar para todo el espectáculo, no hace falta cargar un swf en cada escena. En este caso, en vez de un nombre de archivo ponemos un título, aprovechando el hecho de que este nombre (ya sea nombre de archivo o simple título) será visualizado siempre en la intrefaz.

En la comunicación **de Flash a POL** no existe un mensaje de cambio de escena, pero en POL Setup se puede crear una regla global para que una determinada **variable** provoque el cambio de escena (a siguiente o anterior). Para ello tendremos que poner en la interfaz unos botones para el cambio de escena, asignarles nombres de variables, y poner los eventos correspondientes en POL. En la interfaz de ejemplo que he creado he incluido dichos botones y los he llamado **nextScene** y **prevScene.** Es importante recordar que estas son para POL dos **variables como cualquier otra**, y que es necesario poner explícitamente las reglas globales en POL Setup para asignar a estas variables el cambio de escena.

4.2 Variables especiales

Existen variables especiales que, enviadas desde POL a MoviePlayAIR, cambian aspectos de la configuración o del comportamiento. Estas variables se distinguen de las variables comunes simplemente por sus nombres. Son variables a las cuales se ha asignado un significado especial. Existen variables especiales para las siguientes tareas:

- activar o desactivar el envío por parte de MoviePlayAIR de algunas de las variables generadas por el micrófono y el acelerómetro. Ver capítulo sobre micrófono y acelerómetro. Estas variables tienen nombres que empiezan por `micro` o `accel`
- ajustar el umbral de detección del micrófono o del acelerómetro: `microThreshold` y `accelThreshold`
- ajustar el tiempo de debounce del gate del micrófono o del acelerómetro `microDebounce`, `accelDebounce`.
- mutear por completo los envíos de acelerómetro o micrófono: `micoMute`, `accelMute`
- cambiar el modo de funcionamiento de un botón (push, toggle, pushStrict) o de un slider o xy (steady, jump), o hacer el controlador visible o invisible. Ver capítulo sobre customización de la interfaz. Estas variables tienen nombres que empiezan por guión bajo.

Podemos enviar estas variables desde POL para cambiar la configuración o el comportamiento del dispositivo a lo largo del espectáculo; o podemos poner en la interfaz customizada controles (botones, sliders) para estas variables de manera que las podremos controlar a mano desde el propio dispositivo.

5 Acelerómetro y micrófono

Los valores generados por el acelerómetro y el micrófono se envían a Pol como si fueran unas variables más, con unos nombres preestablecidos que no se pueden cambiar.

5.1 Acelerómetro

El acelerómetro envía por defecto las siguientes variables:

- **`accelX`**, **`accelY`**, **`accelZ`**: son los tres valores "crudos" tal como los mide el acelerómetro y corresponden a las tres proyecciones de la aceleración sufrida por el dispositivo en los tres ejes del espacio. Recordemos que el campo gravitacional es percibido como una aceleración de 1g hacia arriba (dirección positiva del eje Z si el dispositivo está tumbado con la pantalla hacia arriba) y es indistinguible de una aceleración debida a un movimiento. El rango de estos tres valores depende del dispositivo pero por lo general debe ser de -3 a 3 o de -5 a 5; en todo caso se mide en g y siempre puede ser positivo o negativo.

- **`accelHpStr`**: "Hp" significa HighPass y "Str" significa "Strength". Este valor es el resultado de un cálculo hecho por la aplicación a partir de los tres anteriores con lo cual no ofrece ninguna información añadida pero es útil puesto que en Pol no se pueden hacer cálculos aritméticos complejos. HighPass significa que elimina la componente continua de la aceleración, es decir, sólo nos da las variaciones con respecto al valor medio. Esto significa que independientemente de la orientación del dispositivo (tanto si está en vertical, en horizontal, boca arriba, boca abajo) este valor siempre nos permite detectar los movimientos y nos da 0 cuando el dispositivo no se mueve, esté como esté. Strength significa que representa la magnitud de la aceleración independientemente de su dirección: es decir es la combinación de las tres componentes x,y,z. El valor es siempre positivo (o nulo) y se mide en g, con lo cual el máximo que puede alcanzar será aproximadamente 3 o 5 dependiendo del dispositivo.

- **`accelPitch`**, **`accelRoll`**: se calculan a partir de accelX accelY y accelZ y representan la orientación del dispositivo en términos de "cabeceo" y "balanceo", que corresponden a dos rotaciones del dispositivo en torno a dos de sus ejes. El ángulo de "viraje" ("yaw") no se puede calcular, ya que sea cual sea ese ángulo el acelerómetro percibe las mismas aceleraciones. Pensemos que si yo estoy de pie noto la gravedad que tira hacia mis pies pero no cambia si miro hacia el norte o hacie el este o hacia el sur. Estos valores sólo tienen sentido cuando el dispositivo está más o menos estable; cuando está moviéndose, como la aceleración no se puede distinguir de la gravedad, los resultados son una mezcla de las dos cosas sin mucho sentido. Por otro lado, **el cálculo trigonométrico que hago para calcular estos ángulos no es el correcto** aunque se acerca lo suficiente como para dar el pego. No es que no se pueda calcular correctamente, es que no he dado con la fórmula correcta y mis nociones de trigonometría están algo oxidadas. El rango es de -180 a 180, los ángulos están en grados.

- **`accelGate`**: es una variable **on/off** que indica si el valor de accelHpStr supera un cierto umbral. El umbral por defecto es 0.5 y se puede cambiar enviando desde Pol una variable especial llamada **`accelThreshold`** o desde la misma interfaz si en ésta incluimos un control para una variable llamada `accelThreshold`. Esta variable accelGate no es muy necesaria, ya que podemos poner fácilmente un gate en Pol Setup sobre la variable `accelHpStr`. La ventaja que puede tener es la de poder cambiar el umbral en cualquier momento. Otra ventaja es que este detector lleva incorporado un debouncing, es decir elimina secuencias rápidas encendido-apagado-encendido debidas al ruido si se producen en un tiempo rápido (por defecto 200 milisegundos). Hacerlo en Pol sería un poco complicado.

El acelerómetro calcula más variables que por defecto no se envían pero cuyo envío se puede activar enviando desde Pol unas variables especiales (o desde el Android si incluimos en la interfaz botones para estas variables especiales). Las otras variables que calcula el acelerómetro son:

- **`accelHpX`**, **`accelHpY`**, **`accelHpZ`**: Hp significa HighPass. Son las tres proyecciones de

la aceleración sin la componente continua, es decir, las variaciones con respecto al valor medio. Constituyen las tres componentes en los tres ejes de accelHpStr y sirven para lo mismo (detectar movimientos eliminando la gravedad) pero por separado eje por eje. De esta manera podemos hacer que el sistema reaccione de manera diferente si sacudimos el dispositivo en sentido horizontal o en sentido vertical.

- **`accelLpX`**, **`accelLpY`**, **`accelLpZ`**: son lo mismo que accelX, accelY y accelZ pero pasados por un filtro pasabajo (LowPass), que es exactamente lo opuesto al HighPass. Se elimina el ruido y se obtiene una señal suavizada (y como efecto secundario, retrasada). Pueden estar bien para conocer la orientación del dispositivo eliminando el temblor, por ejemplo cuando lo tenemos en las manos. Sin embargo los valores "crudos" ya vienen bastante filtrados desde el hardware, por lo tanto es probable que estas variables no sirvan de mucho. Por esto están deshabilitadas por defecto.
- **`accelLpPitch`** y **`accelLpRoll`**: son los ángulos de orientación pero filtrados con el filtro LowPass. Valen las mismas consideraciones que se han hecho sobre las tres variables anteriores, en este caso con respecto a los ángulos. También en este caso, es probable que no resulten muy útiles puesto que los valores sin filtrar ya vienen bastante filtrados desde el hardware. No existen ángulos "high pass" ya que los valores que se obtendrían serían totalmente ficticios y no representarían ningún ángulo real, porque el cálculo del ángulo asume que la aceleración que se mide sea debida a la gravedad, mientras que las variaciones bruscas de aceleración no son debidas a la gravedad.

5.1.1 Variables especiales para controlar los envíos del acelerómetro

Las siguientes variables especiales sirven para activar y desactivar el envío de las variables mencionadas anteriormente. Véase también el capítulo sobre variables especiales. Nota: en la interfaz de ejemplo que he preparado no he incluido botones para estas variables especiales, excepto para la variable accelMute. Todas estas son variables on/off: 1 significa activar y 0 desactivar:

- **`accelSendXYZInput`**: activa o desactiva el envío de accelX accelY y accelZ (crudos). Activado por defecto
- **`accelSendXYZLowPass`**: activa o desactiva el envío de accelLpX, accelLpY, accelLpZ. Desactivado por defecto
- **`accelSendXYZHighPass`**: activa o desactiva el envío de accelHpX, accelHpY, accelHpZ. Desactivado por defecto
- **`accelSendStrengthHighPass`**: activa o desactiva el envío de accelHpStr. Activado por defecto.
- **`accelSendGate`**: activa o desactiva el envío de accelGate. Activado por defecto
- **`accelSendRawAngles`**: activa o desactiva el envío de los ángulos accelPitch y accelRoll (sin filtrar). Activado por defecto
- **`accelSendLowPassAngles`**: activa o desactiva el envío de los ángulos filtrados accelLpPitch y accelLpRoll. Desactivado por defecto
- **`accelMute`**: mutea todos los envíos de acelerómetro, independientemente de los valores de las variables anteriores. El valor 1 significa mutear (dejar de enviar) y 0 significa desmutear (volver a enviar). Cuando desmuteamos, se volverán a enviar aquellas variables cuyos envíos estaban activados.

Existen también las siguientes dos variables especiales de rango:

- **`accelThreshold`**: umbral de detección del gate `accelGate`. El valor por defecto es 0.5.
- **`accelDebounce`**: tiempo de debounce del gate: durante cuánto tiempo de una transición

on/off o off/on se inhiben otras transiciones. Valor por defecto 200, se mide en milisegundos.

Como con todas las variables especiales, es posible incluir en la interfaz controladores para controlar estas variables manualmente en vez de enviarlas desde Pol. En la interfaz de ejemplo he puesto un slider para el threshold pero no para el tiempo de debounce.

5.2 Micrófono

Funciona de manera análoga al acelerómetro aunque es mucho más simple porque sólo genera dos variables:

- **`microValue`**: el valor de intensidad de sonido detectada por el micrófono, en el rango de 0 a 100.
- **`microGate`**: variable on/off que indica si el valor de `microValue` supera o no un determinado umbral que por defecto es 50. Igual que el gate de acelerómetro, no es muy necesario puesto que se puede poner en Pol una regla de tipo gate sobre la variable `microValue`; la ventaja de detectar y enviar el gate desde el dispositivo puede ser la de poder cambiar en cualquier momento el umbral y el hecho de que tiene incorporado un debouncing.

El envío de ambas estas variables está activado por defecto.

5.2.1 Variables especiales para controlar los envíos del micrófono

Las siguientes variables especiales on/off sirven para activar/desactivar los envíos de micro:

- **`microSendValue`**: activa/desactiva el envío de `microValue`. Activo por defecto
- **`microSendGate`**: activa/desactiva el envío de `microGate`. Activo por defecto
- **`microMute`**: mutea/desmutea ambos envíos de micro. 1 mutea, 0 desmutea. Cuando desmuteamos, se vuelven a enviar solo las variables cuyo envío estaba activado.

Al tratarse sólo de dos variables, es discutible la utilidad de `microSendValue` y `microSendGate` ya que podemos dejar activadas las dos siempre y mutear las dos con `microMute` si necesario. Sin embargo, si sólo vamos a usar el gate y no el valor, puede ser interesante limitar el tráfico en la red evitando el envío tan frecuente de un valor innecesario y enviar sólo el gate.

En la interfaz de ejemplo he puesto un botón para `microMute` pero no para las otras dos, pero como todas las variables especiales, es posible incluir en la interfaz botones para todas.

Existen también las siguientes variables especiales de rango:

- **`microThreshold`**: el umbral de detección del micro. Valor por defecto 50.
- **`microDebounce`**: el tiempo de debounce del gate del micro en milisegundos. Por defecto es 200.

En la interfaz de ejemplo he incluido un slider para el threshold.

6 Customización de la interfaz

Customizar la interfaz significa no solamente cambiar el aspecto gráfico y la disposición de los botones y controles, sino también decidir cuáles y cuántas variables se envían y se reciben, de qué tipo (on/off, rango), con qué nombres y afinar parámetros como el valor mínimo y máximo de un rango, etc.

Esto se hace **editando archivos fla y generando los correspondientes swf**. No es necesaria ni una línea de código ActionScript para customizar la interfaz y decidir cuáles y cuántas variables tiene que haber y de qué tipo en todo el espectáculo o en una escena concreta: se trata de incluir en el fla *movie clips* respetando unas nomenclaturas y estructuras rigurosas para que la aplicación reconozca los elementos y les asigne los comportamientos requeridos. El fla no necesita estar asociado a un archivo .as: no es que no haga falta tocar la programación, es que *no lleva* programación.

Sin embargo se puede incluir código para añadir comportamientos arbitrarios que se salen de los previstos.

El archivo **interface.swf** es el que contiene la **interfaz general**. Editando este archivo podemos:

- decidir cuántas y cuáles **variables** y de qué tipo se enviarán (y recibirán) desde el dispositivo. Estas variables existirán **durante todo el espectáculo** (pero esto no significa en absoluto que tengan que corresponder a reglas globales en POL: de la misma manera que habitualmente asignamos en cada escena eventos distintos a cada botón y eje del Joydreske, así podremos asignar a cada variable del Android comportamentos distintos en cada escena; eso se hace en POL Setup).
- **diseñar** la disposición y el aspecto de estos controles y de los “displays” que visualizan el número y nombre de escena, los monitores que muestran el estado de acelerómetro y micrófono, etc.
- decidir dónde se coloca el **swf de la escena** (si existe) que será cargado en cada escena.
- disponer todos estos elementos en un espacio que excede la pantalla, y que se puede “hojear” por **páginas**, y decidir cuáles de estos elementos están anclados a las páginas y cuáles en cambio están anclados a la pantalla y por lo tanto son siempre visibles e inamovibles.

Si se modifica el interface.fla y se genera una nueva versión de interface.swf, **no es necesario reinstalar la apliación**: simplemente hay que reemplazar interface.swf en la carpeta “pol” en Documentos.

Los archivos swf individuales de cada escena funcionan de la misma manera, pero el único tipo de elementos de interfaz (movieclips con nombres especiales) que se pueden poner dentro de ellos son los de las **variables**. Las convenciones de nomenclatura son las mismas que para el archivo interface.swf. Recordemos que los nombres de los archivos swf de escena tienen que corresponder con los nombres que se configuran en POL Setup haciendo doble click en la escena, y que estos archivos serán cargados de uno en uno (nunca coexistirá más de uno) dentro de un contenedor que se encuentra en interface.swf, o en su defecto, directamente dentro de interface.swf por en cima de todo.

6.1 Nomenclatura y estructura

Lo que permite a la aplicación reconocer los elementos de la interfaz y asignarles los **comportamientos** debidos, y crear las **variables** en el caso de botones y controles, son los **nombres de instancia** de los movieclips y la **estructura**, en el sentido de quién es hijo de quién (se dice que un movie clip es hijo de otro si está contenido en él). Subrayo nombre **de instancia** porque lo que cuenta es el nombre de la instancia en el escenario, no el nombre que tiene el símbolo en la librería/biblioteca (es un despiste común confundir una cosa por la otra).

La **estructura** es por lo general **bastante flexible**, en el sentido de que un determinado elemento funcionará correctamente esté donde esté mientras tenga el nombre de instancia que tiene que tener - con pocas excepciones (que serán documentadas) en las que tal elemento tiene que ser hijo directo de tal otro elemento para ser reconocido.

Al nivel más externo, todos elementos relevantes para el funcionamiento de la interfaz tienen **nombres de instancia que empiezan por "__POL"** (con dos guiones bajos). Algunos de ellos requieren a su vez que dentro de ellos haya hijos con nombres determinados (como p.e. el cursor de un fader), que en este caso empezarán también con dos guiones bajos "__" pero sin "POL".

6.1.1 Elementos especiales de interfaz (interface.swf)

Los elementos especiales de interfaz que se pueden encontrar en interface.swf son los siguientes. Ninguno de ellos es obligatorio, todos pueden ser ausentes, en cuyo caso se renunciará a la funcionalidad que ofrecen:

- **`__POL_paged`**: es el contenedor de las páginas. Puede estar a cualquier nivel de la gerarquía de movieclips. Todo lo que se encuentra dentro de este movieclip se podrá hojear por páginas (corriendo el dedo sobre la pantalla) mientras que todo lo que esté fuera de él se quedará siempre anclado a la pantalla. Decidimos nosotros qué poner dentro de pages y qué dejar fuera, dependiendo de si queremos que un elemento se quede siempre en la pantalla o no. Más detalles en el capítulo específico.
    - **`__sens`** (hijo directo, opcional): si el clip `__POL_paged` tiene un hijo directo llamado `__sens`, este será utilizado para determinar el área efectiva de la interfaz a fines de la paginación; si este clip es ausente, se utilizará el área total real ocupada por todo el clip `__POL_paged`

- **`__POL_part_display`**: es el display que muestra el número de escena y el nombre de escena o del archivo swf cargado (ver la sección "cambio de escena" anterior). Tiene que contener (no necesariamente hijos directos sino descendientes) dos campos de texto:
    - **`__scene_number`**: tiene que ser un **campo de texto dinámico**. En él se mostrará el número de escena actual. Tiene que estar dentro de `__POL_part_display` pero no hace falta que sea hijo directo, puede ser hijo de un hijo etc.
    - **`__file_name`**: tiene que ser un **campo de texto dinámico**. En él se mostrará el nombre de archivo de escena tal como está configurado en POL Setup para la escena actual. Si utilizamos archivos swf será el nombre del archivo cargado; si no utilizamos archivos swf de parte, podemos utilizarlo para poner un título de escena. En todo caso se visualizará tal como viene de Pol. El campo de texto tiene que estar dentro de `__POL_part_display` pero no hace falta que sea hijo directo, puede ser hijo de un hijo etc.

- **`__POL_part_swf_container`**: es el contenedor dentro del cual se cargará el swf individual de cada escena. Lo normal es que sea un movieclip vacío. Puede estar dentro o fuera del `__POL_paged` (como todo). **Si no existe este** clip **no** significa que no se carguen los swf de escena, sino que se cargarán igualmente y se pondrán "sueltos" en la raíz del escenario, en posición (0,0), y por en cima de todo lo demás, y por supuesto fuera de las páginas.

Estos tres elementos tienen que ser **únicos**: no puede haber más de un clip con el mismo nombre; de lo contrario sólo uno de ellos tendrá la funcionalidad esperada y puede que no funcione del todo correctamente.

6.1.2 Controladores para variables (interface.swf y/o swf de escena)

Los controladores que envían variables (botones, faders, touchpads) son movieclips cuyo nombre de instancia empieza por "**`__POL_control_`**" y tienen que respetar una estructura interna al clip que será detallada a continuación en función del tipo de controlador. Los controladores para variables pueden encontrarse tanto en interface.swf como en los swf individuales de las escenas. Un controlador **envía** la variable que tiene asignada cuando interactuamos con él, y también **visualiza** el valor de la variable si se recibe esta variable desde Pol. El **nombre de la variable** asignada al controlador se deduce del **nombre de instancia** del clip, junto con otros parámetros.

De momento existen tres tipos de controlador:

- **botón:** para variable de tipo **on/off**
- **slider (o fader):** para variable de **rango**
- **touchpad (o xy)**: para controlar **dos variables de rango** al mismo tiempo.

6.1.3 Botones

El nombre de instancia de un clip para que funcione de botón tiene que respetar la sintaxis:

- **`__POL_control_onoff_`*`modo_nombrevariable`*** o en alternativa:
- **`__POL_control_onoff_`*`nombrevariable`***

donde *nombrevariable* tiene que ser reemplazado por el nombre de la variable que se quiere enviar y recibir, y *modo* tiene que ser reemplazado por uno de los tres modos de funcionamiento siguientes.

Los **modos** de funcionamiento posibles de un botón son tres:

- **`pushStrict`**: se trata de un botón clásico propiamente dicho. Cada vez que apretamos el botón, se envía la variable con valor 1 y el botón se enciende; cuando soltamos el botón, se envía la variable con valor 0 y se apaga el botón. Si apretamos, arrastramos el dedo afuera y soltamos es como si soltáramos normalmente: nunca se queda "enganchado" el botón.
- **`toggle`**: se trata de un interruptor. Cuando lo apretamos cambia de estado: si estaba apagado se enciende y si estaba encendido se apaga, y envía en cada caso el valor correspondiente (1 para encendido y 0 para apagado). Siempre se queda en el último estado independientemente de cómo soltamos el dedo.
- **`push`**: nos permite usar el mismo botón como push o como toggle según necesidad. Cuando apretamos el botón se enciende y envía 1, a no ser que estuviera ya encendido. Si lo soltamos normalmente, con el dedo aún dentro de él, se apaga y envía 0 como si fuera un push. Si arrastramos el dedo hacia fuera y soltamos, el botón se queda enganchado en estado encendido y no envía el 0. Para apagarlo lo apretaremos otra vez y esta vez soltaremos el dedo sin arrastrarlo afuera.

Si no se indica el modo en el nombre de instancia, por defecto el botón será de tipo push.

Sea del tipo que sea, el botón cambiará su estado si recibe una variable desde POL con el nombre de variable correspondiente y valor distinto a su estado actual, y así lo visualizará.

Ejemplos de nombres de instancia para botones:

`__POL_control_onoff_button1`: botón para la variable "button1". Será de tipo push por defecto.

`__POL_control_onoff_push_button2`: botón de tipo push para la variable “button2”.

`__POL_control_onoff_toggle_musicaOnOff`: botón de tipo toggle para la variable “musicaOnOff”.

El clip botón tiene que tener dos **hijos directos** con los siguientes nombres de instancia:

- **`__on`**: este clip será visible cuando el botón esté en estado on, e invisible cuando esté en estado off
- **`__off`**: este clip será visible cuando el botón esté en estado off, e invisible cuando esté en estado on.

Si no tiene estos hijos, el botón funcionará igualmente para enviar su variable pero no se podrá ver en qué estado está, ni mucho menos ver si se ha recibido la variable desde Pol.

Dentro y fuera de `__on` y `__off` se pueden poner todos los clips y otros elementos gráficos que hagan falta. Lo que esé fuera de `__on` y `__off` será visible siempre, independientemente del estado del botón.

El modo del botón (push, toggle, pushStrict) se puede cambiar en tiempo real enviando una variable desde Pol. Véase la sección sobre variables especiales.

6.1.4 Sliders

El nombre de instancia de un clip para que funcione de slider tiene que respetar una de las la siguientes sintaxis:

- **`__POL_control_range_`*`modo_nombrevariable_min_max`***
- **`__POL_control_range_`*`modo_nombrevariable`***
- **`__POL_control_range_`*`nombrevariable_min_max`***
- **`__POL_control_range_`*`nombrevariable`***

donde *nombrevariable* tiene que ser reemplazado por el nombre de la variable que se quiere enviar y recibir, y *min* y *max* por los valores mínimo y máximo del rango. Si no se especifica un rango, **el rango por defecto será de 0 a 100**. **El nombre de variable no puede contener guiones bajos**.

Para indicar **valores negativos** para min y max, hay que usar la letra **`m`** en lugar del signo menos antepuesta al número (el signo “-” no se puede usar en nombres de instancias).

El ***modo*** puede ser **`steady`** o **`jump:`**

- **`steady:`** cuando tocamos el slider, el cursor se mueve a partir del punto en el que se encuentra, aunque no coincida con el punto donde está nuestro dedo. De esta manera el slider no puede hacer saltos bruscos: se parece en este sentido a un fader real.
- **`jump`**: cuando tocamos el slider, el cursor salta inmediatamente al punto donde tocamos, con lo cual siempre está pegado a nuestro dedo.

El **modo por defecto**, si no se especifica en el nombre de instancia, es **jump**.

Ejemplos de nombres de instancia para sliders:

`__POL_control_range_ejeX`: slider (modo jump) para una variable llamada “ejeX”, en el rango por defecto de 0 a 100.
`__POL_control_range_ejeY_m50_50`: slider (modo jump) para una variable llamada “ejeY” en el rango de -50 a 50.
`__POL_control_range_steady_videoSpeed_m3_3`: slider (modo steady) para una variable llamada “videoSpeed” en el rango de -3 a 3.

La estructura interna del clip de un slider tiene que respetar unas pautas para que el funcionamiento sea correcto.

- Internamente el slider es **siempre horizontal**: es decir, el cursor se mueve en el eje X, nunca el Y. Para crear un slider vertical, simplemente hay que poner en el escenario un slider rotado de 90º en sentido contrario a las agujas del reloj (si queremos que valores altos estén arriba arriba).

El slider tiene que contener los siguientes tres movieclips (o almenos los primeros dos, el último es opcional) con los siguientes nombres de instancia: no hace falta que sean hijos directos, pueden estar a cualquier nivel de la gerarquía:

- **`__bar`**: determina el espacio en el cual se puede mover el cursor. El cursor sólo se mueve en horizontal, y se moverá desde la extremidad izquierda de `__bar` hasta la extremidad derecha de `__bar`. La izquierda corresponde al valor mínimo del rango y la derecha al valor máximo.
No hay que preocuparse de que el centro o la esquina superior izquierda o ningún punto en concreto del `__bar` coincida con el origen (0,0) del clip ni que su origen esté alineado de alguna manera con el origen del padre. Lo que sí hay que tener en cuenta es que el cursor puede moverse hasta que su propio origen (0,0) esté alineado con la extremidad izquierda o derecha del `__bar`. Si queremos que el cursor sólo llegue a tocar los bordes de la barra con sus propios bordes, tendremos que crear un `__bar` transparente que será un poco más corto que la barra visible, la cual estará fuera de `__bar` y no tendrá ningún nombre especial.
- **`__cursor`**: es el cursor, cuya posición refleja en todo momento el valor de la variable. Su origen (0,0) es la que podrá correr desde estar alineada con una extremidad de `__bar` hasta estar alineada con la otra extremidad de `__bar`. Por ello es aconsejable que el contenido del clip `__cursor` esté centrado en torno a su 0,0 para que sea más fácil posicionar correctamente el cursor con respecto al `__bar`.
- **`__sens`**: si extste, este clip determina el área sensible del slider, es decir, el slider reaccionará al toque si tocamos el área ocupada por `__sens`. Si no existe este clip, el área sensible será la totalidad del área ocupada por todo el clip del slider

En principio no es necesario que `__bar` y `__cursor` sean hijos del mismo padre: todo debería funcionar correctamente incluso aunque `__cursor` esté dentro de un clip distinto al que contiene `__bar` y éstos tengan escalados, posiciones e incluso rotaciones distintos; pero esto no lo he testeado a fondo. Lo que sí no puede ser es que `__bar` esté dentro de `__cursor` (cosa que no tendría ningún sentido), mientras que en principio `__cursor` sí puede estar dentro de __bar aunque no lo recomiendo.

6.1.5 Touchpads tipo “xy”

Un touchpad es un rectángulo o cuadrado con un cursor dentro: tocando el touchpad con el dedo se mueve el cursor en el espacio dentro del rectángulo. Un touchpad controla **dos variables de rango a la vez**: una vinculada con la posición horizontal (X) del cursor y la otra con la posición vertical (Y).

Las convenciones de estructura interna son las mismas que las de un **slider**, con la única diferencia que el cursor se mueve tanto en horizontal como en vertical, dentro del rectángulo marcado por el clip `__bar`: lo que se ha dicho para los sliders sobre la posición X del cursor con respecto a las extremidades izquierda y derecha de `__bar` vale también para la posición Y del cursor con respecto a las extremidades superior e inferior de `__bar`.

El nombre de instancia tiene que respetar una de las sintaxis siguientes:

- **`__POL_control_xy_`*`modo_variable1__variable2`***
- **`__POL_control_xy_`*`variable1__variable2`***

donde cada uno de los bloques variable1 y variable2 puede tener cualquiera de las dos formas:

- ***`nombrevariable`*** sin más
- ***`nombrevariable_min_max`***

Es importante notar que el separador para separar las dos variables es "__": dos guiones bajos. **Los nombres de variable no pueden contener guiones bajos**.

La primera variable es la que está asociada al eje X y la segunda al eje Y.

Los **modos** son los mismos de un slider: **`jump`** y **`steady`**, siendo jump el modo por defecto.

Ejemplos:

`__POL_control_xy_ejeX__ejeY`: controlador touchpad (modo jump) para las variables "ejeX" y "ejeY" las dos en el rango por defecto de 0 a 100.

`__POL_control_xy_steady_pitch_m12_12__volume_0_2`: controlador touchpad (modo steady) para las variables "pitch" y "volume", la primera en el rango de -12 a 12 (podrían ser semitonos) y la segunda en el rango de 0 a 2.

6.1.6 Variables especiales que afectan el comportamiento de botones/controles

Por cada controlador de variable que creemos, existen unas variables especiales asociadas que (enviadas desde POL al Android) afectan el comportamiento de ese controlador. Estas variables especiales tienen nombres que empiezan por guión bajo y que acaban por el nombre de la variable de la que afectan el comportamiento:

- **`_visible_`*`nombrevariable`***: hace visible (valor 1) o invisible (valor 0) el controlador de la variable *nombrevariable*. Creo que no funciona con controladores de tipo "xy" (es un bug)
- **`_pushMode_`*`nombrevariable`***: el valor tiene que ser 1. Cambia el modo del botón *nombrevariable* a modo push.
- **`_pushStrictMode_`*`nombrevariable`***: el valor tiene que ser 1. Cambia el modo del botón *nombrevariable* a modo pushStrict.
- **`_toggleMode_`*`nombrevariable`***: el valor tiene que ser 1. Cambia el modo del botón *nombrevariable* a modo toggle.
- **`_jumpMode_`*`nombrevariable`***: el valor tiene que ser 1. Cambia el modo del slider *nombrevariable* a modo jump. No funciona con controladores de tipo xy, sólo sliders.
- **`_steadyMode_`*`nombrevariable`***: el valor tiene que ser 1. Cambia el modo del slider *nombrevariable* a modo steady. No funciona con controladores de tipo xy, sólo sliders.

6.1.7 Monitores para los valores de acelerómetro y micrófono

Los valores generados por acelerómetro y micrófono en realidad no son más que variables como cualquier otra, con unos nombres de variable preestablecidos. Los elementos de interfaz que visualizan estos valores son en realidad **sliders y botones** como cualquier otro botón o slider sólo que en vez de reaccionar al toque, reaccionan a los movimientos del acelerómetro y al nivel del micrófono. Por lo tanto, para crear estos monitores en la interfaz tenemos que crear movieclips con las mismas convenciones que seguiríamos para crear sliders o botones para variables con esos

nombres. Los nombres de las variables del acelerómetro y del micro están listados en el capítulo correspondiente.

Por ejemplo, el valor de aceleración X tiene asignado el nombre de variable “accelX” y por lo tanto si ponemos un movieclip llamado `__POL_control_range_accelX_m3_3` tendremos automáticamente un monitor que nos mostrará el valor de aceleración X en el rango de -3 a 3. También podríamos decidir poner un display de tipo xy para las aceleraciones x e y poniendo un controlador xy con nombre de instancia `__POL_control_xy_accelX_m3_3__accelY_m3_3`.

El hecho de que estos controladores-monitores estén presentes o no en la interfaz **no** influye en que se envíen o no los valores correspondientes a POL. Los valores se enviarán a Pol igualmente; los monitores los pondremos o no dependiendo de si tenemos necesidad de monitorizar estos valores.

Estos monitores se pueden poner tanto en la interfaz general interface.swf como en el swf individual de una determinada escena, y no hay problema en que haya duplicados de alguno de estos monitores.

Para monitorizar los ángulos del acelerómetro (véase el capítulo sobre el acelerómetro) existe un tipo de controlador que se llama “angles”; al igual que un controlador de tipo “xy”, visualiza dos valores a la vez, pero en vez que en coordenadas cartesianas los visualiza como ángulos de *pitch* y *roll* (cabeceo y balanceo) de un objeto en 3D. Los nombres de las variables de los ángulos del acelerómetro son accelPitch y accelRoll y el nombre de instancia del monitor de ángulos por lo tanto es `__POL_control_angles_accelPitch_accelRoll.`

La programación de este monitor está bastante verde y en vez de intentar documentarlo prefiero dejar una instancia en la interfaz de ejemplo para copiar y pegar.

6.1.8 Controladores para variables especiales

Como hemos visto anteriormente existen **variables especiales** que sirven para cambiar el comportamiento o el estado de la aplicación. Ejemplos de variables especiales son:

- las que activan o desactivan los envíos de acelerómetro y micrófono
- las que determinan el umbral o el tiempo de debounce del gate de micrófono o de acelerómetro
- las que hacen visible o invisible el controlador (botón/slider) de otra variable
- las que cambian el modo de funcionamiento de un botón (entre push y toggle o pushStrict) o de un slider (entre steady y jump).

Las últimas dos categorías a lo mejor son de discutible utilidad en lo inmediato, pero las primeras dos sin duda son necesarias.

Estas variables se pueden enviar desde POL para configurar el comportamiento del dispositivo. Pero también podemos poner controladores para estas variables en la propia interfaz. Esto puede servir para dos cosas:

- monitorizar desde el dispositivo el valor de estas variables y saber en todo momento cuál es
- cambiar el valor de estas variables manualmente desde el dispositivo en vez que desde POL: por ejemplo, mutear el envío de micrófono, o ajustar un umbral a mano en el momento.

Para poner controladores para estas variables especiales en la interfaz se sigue exactamente el mismo mecanismo que para las variables comunes.

Por ejemplo, las variables especiales `microMute` y `accelMute` sirven para mutear todos los envíos de micrófono y acelerómetro: si queremos poner unos botones en la interfaz para mutear el

micro y el acelerómetro, tendrán que llamarse como si fueran botones para enviar estas variables a Pol: los nombres de los dos botones serán algo así como `__POL_control_onoff_toggle_microMute` y `__POL_control_onoff_toggle_accelMute` (el modo toggle es el más adecuado en este caso).

Otro ejemplo: para poner un slider que controle el umbral del micrófono, que se llama microThreshold, pondremos un slider con nombre de instancia `__POL_control_range_microThreshold_0_100` (la terminación en `_0_100` de hecho no es necesaria puesto que es el rango por defecto).

Si ponemos estos controles, podremos controlar estas variables (que afectan el comportamiento del dispositivo) desde el dispositivo mismo, pero en cualquier caso **estas variables nunca se enviarán a POL**.

6.2 Páginas

En la mayoría de los casos, no habrá suficiente espacio en la pantalla para todos los controles deseados. Por ello, el sistema permite organizar los elementos de la interfaz en un **espacio más amplio que la pantalla**. A la hora de usar la interfaz, se accederá a los controles que estén fuera de la pantalla haciendo el gesto de "hojear" típico de los dispositivos táctiles, que consiste en tocar la pantalla y arrastrar el dedo hacia un lado o hacia otro. Todos los controles se desplazarán hacia el lado indicado, de manera que los que antes estaban dentro de la pantalla saldrán y otros entrarán. La medida del desplazamiento siempre será exactamente igual al ancho de la pantalla.

Es importante entender que, aunque esto produzca la ilusión de "hojear" una serie de página, no hay realmente ninguna separación física entre una página y otra. Lo que hay es una "**butifarra**" continua que puede extenderse todo lo que queramos pero que siempre se desplaza por incrementos múltiples del ancho de la pantalla hasta llegar a cualquiera de sus extremidades. Por ello, la interfaz tiene que estar diseñada **teniendo en cuenta la resolución de la pantalla**, y una interfaz diseñada para una resolución no será adecuada para un dispositivo que tiene otra resolución. Es un sistema susceptible de muchas mejoras.

Este sistema soporta tanto páginas organizadas en horizontal (butifarra horizontal) como páginas organizadas en vertical (butifarra vertical) pero no las dos cosas a la vez. El sistema detecta automáticamente si la interfaz excede la pantalla más en sentido horizontal o más en sentido vertical y actúa en consecuencia.

La paginación se aplica al clip llamado **`__POL_paged`**: todo lo que esté fuera de este clip no se moverá y siempre quedará dentro de la pantalla. Podemos decidir cuáles elementos de la interfaz poner dentro de `__POL_paged` y cuáles no, en funcion de las necesidades. En otras palabras el clip `__POL_paged` *es* la butifarra.

El clip `__POL_paged` puede tener un **hijo directo** con nombre de instancia **`__sens`**, que marca el área que se tendrá en cuenta a la hora de hojear las páginas. Es decir, si existe un clip `__sens` dentro de `__POL_paged`, será el área ocupada por `__sens` la que se considerará como área de la butifarra, y no el área real. Esto puede servir si tenemos un fondo que excede el área visible y queremos evitar que la parte excedente afecte el cómputo de las páginas: en este caso pondremos un `__sens` transparente con las medidas exactas. Si falta el clip `__sens`, el sistema considerará el área real de todo `__POL_paged`.

El área de `__POL_paged` puede exceder la pantalla tanto hacia la derecha como hacia la izquierda. O tanto hacia arriba como hacia abajo en el caso de butifarra vertical. Es decir, que la página visualizada inicialmente por defecto (que es la que cabe dentro del escenario en el fla) no tiene por qué ser la primera de la izquierda. Puede haber páginas más a la izquierda.

Como cualquier otro elemento, el contenedor para el swf de escena, **`__POL_swf_part_container`**, puede estar dentro o fuera de `__POL_paged`, dependiendo de si queremos que los controles específicos de escena estén anclados a la pantalla y siempre visibles, o anclados a una página en concreto .

Por ejemplo, el Galaxy Tab tiene una resolución de pantalla de 1024x600. En la interfaz de ejemplo he puesto los botones button1-button10 en la página central, junto con un controlador xy que envía las variables “ejeX” y “ejeY”. A la izquierda de esta página he puesto una página que contendrá el swf de escena, y a la derecha una página con los monitores de acelerómetro y micrófono. Para ello, he creado un clip `__POL_paged` con dentro un clip rectangular `__sens` de tamaño 3072x600 (3 veces 1024 de ancho) y lo he colocado de tal manera que se extienda 1024 pixels por fuera de cada lado del escenario.

El clip `__POL_part_swf_container` (que es el contenedor destinado a contener el swf de escena y es un clip vacío) está dentro de `__POL_paged` y se encuentra en -1024,0.

En cambio, el clip `__POL_part_display` y los botones para las variables de cambio de escena nextPart y prevPart, los he puesto fuera de `__POL_paged`, de manera que quedarán siempre visibles en la parte de arriba de la pantalla aunque cambiemos de página. Todas estas son decisiones arbitrarias que pueden ser más convenientes o menos según las necesidades del espectáculo. El hecho de tener los botones de cambio de parte siempre disponibles, por ejemplo, es cómodo pero a la vez peligroso. Puede ser interesante ponerlos en una página a parte para que no sea demasiado fácil cambiar de escena sin querer. Recordemos que los botones para el cambio de escena (que de hecho son dos botones para dos variables como cualquier otra) no tienen por qué estar pegados al `__POL_part_display`: podrían los botones estar en una página y el display estar suelto fuera del `__POL_paged` y siempre visible.

7 Resumen de variables

7.1 Variables predefinidas

- `microValue` *rango 0..100*
- `microGate` *on/off*
- `accelX` *rango, g*
- `accelY` *rango, g*
- `accelZ` *rango, g*
- `accelHpStr` *rango, g*
- `accelGate` *on/off*
- `accelPitch` *rango -180..180*
- `accelRoll` *rango -180..180*
- `nextPart` *on/off*
- `prevPart` on/off

7.2 Variables especiales

Si no se indica otra cosa, el valor es 0 (off) o 1 (on).
Micrófono y acelerómetro:

- `accelMute`
- `accelSendXYZInput`
- `accelSendXYZLowPass`
- `accelSendXYZHighPass`
- `accelSendStrengthHighPassz`
- `accelSendGate`
- `accelSendRawAngles`
- `accelSendLowPassAngles`
- `accelDebounce` *(valor en milisegundos)*
- `accelThreshold` *(valor en g)*
- `microMute`
- `microSendValue`
- `microSendGate`
- `microDebounce` *(valor en milisegundos)*
- `microThreshold` *(valor 0..100)*

Relativas a una variable:

- `_visible_`*`nombrevariable`*
- `_pushMode_`*`nombrevariable`* `1`
- `_pushStrictMode_`*`nombrevariable`* `1`
- `_toggleMode_`*`nombrevariable`* `1`
- `_steadyMode_`*`nombrevariable`* `1`
- `_jumpMode_`*`nombrevariable`* `1`

8 Resumen de nomenclatura (movieclips)

8.1 Elementos especiales de interfaz

- **__POL_paged**: contenedor páginas
  - **__sens** (hijo directo, opcional): área sensible
- **__POL_part_display**:
  - **__scene_number**: (descendiente, campo de texto dinámico)
  - **__file_name**: (descendiente, campo de texto dinámico).
- **__POL_part_swf_container**: contenedor para swf de escena.

8.2 Controladores para variables

botón:

- **__POL_control_onoff_*modo_nombrevariable***:

donde *modo* es `push`, `toggle` o `pushStrict`.
__POL_control_onoff_*nombrevariable*:
modo push por defecto

- **__on** (hijo directo)
- **__off** (hijo directo)

slider:

- **__POL_control_range_*modo_nombrevariable_min_max***

donde *modo* es `steady` o `jump`. En *min* y *max* usar la "m" en lugar de "-" para números negativos.
__POL_control_range_*modo_nombrevariable*
por defecto min=0 y max=100.
__POL_control_range_*nombrevariable_min_max*
por defecto modo=`jump`.
__POL_control_range_*nombrevariable*

- **__bar** (descendiente)
- **__cursor** (descendiente)
- **__sens** (descendiente, opcional)

touch xy:

- **__POL_control_xy_*modo_nomvar1_min_max__nomvar2_min_max***

__POL_control_xy_*modo_nombrevariable1__nombrevariable2*
__POL_control_xy_*nomvar1_min_max__nomvar2_min_max*
__POL_control_xy_*nombrevariable1__nombrevariable2*

- **__bar** (descendiente)
- **__cursor** (descendiente)
- **__sens** (descendiente,opcional)

8.3 Monitores y controles especiales

Estos se deducen de los listados anteriores, pero por comodidad de copy-paste pongo aquí los nombres listos y con las configuraciones más razonables (aunque no son las únicas válidas):

- `__POL_control_range_microValue`
- `__POL_control_onoff_microGate`
- `__POL_control_range_accelX_m3_3`
- `__POL_control_range_accelY_m3_3`
- `__POL_control_range_accelZ_m3_3`
- `__POL_control_accelHpStr_0_3`
- `__POL_control_onoff_accelGate`
- `__POL_control_angles_accelPitch_accelRoll`

- `__POL_control_onoff_pushStrict_extPart`
- `__POL_control_onoff_pushStrict_prevPart`
- `__POL_control_onoff_toggle_accelMute`
- `__POL_control_onoff_toggle_accelSendXYZInput`
- `__POL_control_onoff_toggle_accelSendXYZLowPass`
- `__POL_control_onoff_toggle_accelSendXYZHighPass`
- `__POL_control_onoff_toggle_accelSendStrengthHighPass`
- `__POL_control_onoff_toggle_accelSendGate`
- `__POL_control_onoff_toggle_accelSendRawAngles`
- `__POL_control_onoff_toggle_accelSendLowPassAngles`
- `__POL_control_range_accelThreshold_0_3`
- `__POL_control_onoff_toggle_microMute`
- `__POL_control_onoff_toggle_microSendValue`
- `__POL_control_onoff_toggle_microSendGate`
- `__POL_control_range_microThreshold`