
Hybrid Live OF Sketching IDE

GAmuza. Hybrid Live OF Sketching IDE

rel. 043x

GAmuza. Hybrid live coding/modular application. rel 04

Texto: Emanuele Mazza && María José Martínez de Pisón

Dibujos: Carlos Maiques

Valencia, España.

Contenidos

1. ¿Qué es GAmuza? 7

1.1. Configuración Preferencias 11

2. Descripción de la interface 15

2.1. IDE de programación 15

2.2. Barra de Menú 17

3. Interface Módulo Timeline 21

4. Interface Módulo Audio Streaming 24

5. Interface Panel Tracking video (Cameras) 27

5.1. Descripción panel 30

6. Interface Módulo Arduino 36

1. ¿Qué es GAmuza?

Premisas:

Dilema GUI versus Code

En el campo de la enseñanza del arte digital es común ver posiciones que defienden el uso de software libre, pero entre ellos se plantea otro dilema: utilizar programación gráfica, como Pure data, o enseñar código, línea de comandos, como Processing.

Hay un complicado entramado de relaciones y significancias entre ambas posturas que a veces parecen responder a campañas de índole casi publicitaria: gráficos-amigables versus entornos de programación-hostil. El adjetivo tras el guión indica el tono y posicionamiento del discurso. De nuevo el debate entre imagen y texto (textual turn[1] vs. pictorial turn[2]), y como hemos señalado antes, GAmuza opta por la hibridación.

Las interfaces gráficas de usuario (GUI) de los módulos y paneles están programadas y diseñadas para facilitar procesos de comunicación con dispositivos de audio, vídeo o Arduino, pero no para ocultar conceptos o estructuras de programación, por eso incorpora una IDE de programación, estableciendo una situación híbrida que aúna GUI y Code.

Facilitar, para que estudiantes, artistas, o gente interesada no experta, pierda el miedo a algo que comúnmente se califica como demasiado complicado. Pero lo complicado o difícil es solo lo que no se comprende. Por eso….

Tecnología para la gente

…GAmuza no oculta los conceptos o estructuras. Comprender las herramientas que usamos todos los días es el primer paso para no ser esclavo de la tecnología.

Más allá del tipo de relación que los estudiantes de arte mantienen con las tecnologías de información, las personas en general está multiplicando el tiempo y los tipos de usos con estos medios.

> Se ha hecho popular publicar ideas personales, intereses, conversaciones, gustos, opiniones, fotos, videos… Todo el mundo está en línea, desde su portátil o Smartphone, y en 5 minutos te hacen sentir que eres el director de todas tus ficciones o vidas virtuales diferentes, constantemente conectado con todos tus amigos, y lo puedes hacer sin conocer la tecnología que estás utilizando, y esto significa un montón de cosas, una es "gente para la tecnología", o en otras palabras, los

1 Rosalind Krauss (1966), "Welkome to the Cultural Revolution" en ***October***, Vol. 77 [artículo on-line] <http://www9.georgetown.edu/faculty/irvinem/theory/Krauss-October-77-1996-WelcomeToTheCulturalRevolution.pdf > [30.08.2012]

2 Liderado en gran medida por W. J. T. Mitchell <http://humanities.uchicago.edu/faculty/mitchell/home.htm> [30.08.2012]

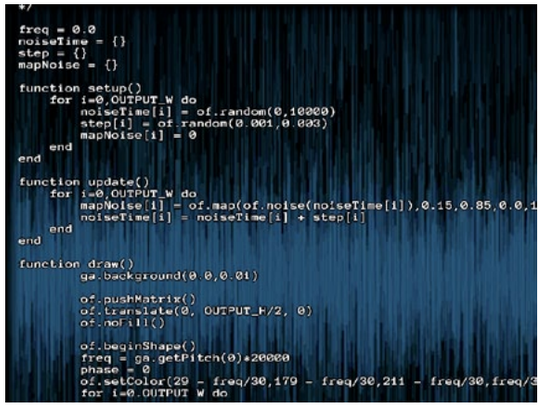
> medios están preparando "gente a la que vender tecnología". Gamuza es sólo un software, pero creado con un concepto en mente: "Tecnología para la gente", no lo contrario.[3]

Conscientes de estas premisas, GAmuza es un software híbrido que conjuga un entorno de programación con distintas aplicaciones modulares para el desarrollo de diseño interactivo, la realización de performances audiovisuales en directo y principalmente la enseñanza de arte interactivo. Es *open-source*, se distribuye bajo licencia MIT[4] y está desarrollado para Linux Ubuntu x86 de 32 o 64 bits y para Mac OSX 10.6 o superior. Puede descargarse en: http://www.gamuza.cc

GAmuza se inició como un software de Live coding para programar en directo, un término vinculado tanto a los entornos de programación con un tiempo de compilación ***infraleve*** —'just in time programming' o 'real time'—, como a la realización en directo de audio y vídeo donde la presencia del código como imagen, o sobre la imagen, es uno de los rasgos que lo caracterizan.

> Los algoritmos son pensamientos, las motosierras son herramientas[5].

Los software de Live Coding, como Fluxus[6] y Zajal[7], fueron los que más influyeron en el desarrollo de GAmuza. Fluxus es un entorno de programación en tiempo real basado en C++ / Scheme y desarrollado por Artem Baguinski. Zajal es un lenguaje de programación creativo, basado en Ruby y que funciona bajo openFrameworks, desarrollado por Ramsey Nasser.



Siguiendo sus pasos, en las primeras versiones de GAmuza, hasta rel.0398, el código se escribía directamente en el módulo de Live Coding, y aun es así en GAmuza para Linux. En la versión 043x para Mac OSX, el código ya no se muestra en la ventana de salida porque cuenta con un editor independiente para facilitar su uso en la enseñanza de programación.

3 Mazza, Emanuele, "Why GAmuza" [texto on-line] <http://gamuza.d3cod3.org/about/concept/> [11.05.2012]
4 MIT License <http://mit-license.org/> [11.05.2012]
5 Stephen Ramsay, Algorithms are Thoughts, Chainsaws are Tools. [video on-line] <https://vimeo.com/9790850> [04.09.2012]
6 (Fluxus) <http://www.pawfal.org/fluxus/> [03.09.2012]
7 Zajal, < http://zajal.cc/> [03.09.2012]

Ahora, el núcleo central de GAmuza es el entorno de desarrollo integrado (IDE integrated development environment) que proporciona un marco de programación muy simplificado con el lenguaje de scripting Lua[8], (utilizando la librería Luabind[9] y una versión modificada de ofxLua[10]) dando así soporte a la preparación de programas que utilicen la librería openFrameworks[11] 0.7.4, gran parte de sus addons[12] y el framework propio de GAmuza que establece conexiones entre estas funciones y los módulos de aplicación GUI, y que permite visualizar las modificaciones del código de forma casi inmediata, sin necesidad de compilación.

Sistema modular y paneles

GAmuza tiene distintos modulos y paneles prediseñados que son fácilmente configurables a través de GUI. Los módulos están siempre activos y los paneles pueden activarse, o no, en función del tipo de trabajo a realizar, para reducir consumo del equipo. Actualmente los módulos son:

- Timeline
- Audio Analysis (input)
- Arduino

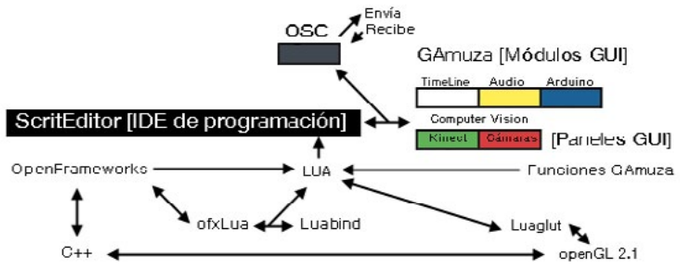
Y los paneles disponibles a través de clases del GAmuza framework:

- Computer vision
- Sensor Kinect

En resumen, GAmuza es un software que recoge y coordina de forma particular lenguajes (Lua y C++) y plataformas (openFrameworks, openCV, openGL, etc.) ya existentes, e incorpora otras librerías propias, para facilitar los primeros pasos de estudiantes y artistas en el campo de la programación creativa. Por otra parte, combina las dos vías habituales de programación, hibridando en el mismo entorno de desarrollo una interface de "code sketching", inspirada directamente en Processing, y modulos gráficos que se ajustan mediante GUI, para que ese proceso de aprendizaje creativo sea más fácil, pero sin ocultar las estructuras de programación. Para hacer funcionar esta hibridación se han programado librerías específicas de GAmuza, que como veremos más adelante, se diferencian sintácticamente de las incorporadas desde otras librerias. Retomando un término de Armando Montesinos, GAmuza más que un collage de lenguajes y plataformas, reconstruye un entorno de programación por medio de "zurcidos", utiliza parches y genera tejido nuevo entrecruzando funciones, sin ocultar los fragmentos.

8 The programming language Lua <http://www.lua.org/> [11.05.2012]
9 Luabind es una librería que ayuda a crear enlaces entre C++ y Lua. Ver http://www.rasterbar.com/products/luabind.html [12.05.2012]
10 OfxLua es un addon de openFrameworks para que funcionen scripts Lua incrustados dentro de una aplicación openFrameworks. Ver https://github.com/danomatika/ofxLua [12.05.2012]
11 openFrameworks <http://www.openframeworks.cc/> [12.05.2012]
12 Directorio de extensiones y librerías para openFramworks http://ofxaddons.com/ [12.05.2012]

La facilidad de uso de GAmuza reside en la simplificación del lenguaje Lua y la comunicación con dispositivos externos a través de los módulos GUI, y todo esto se consigue manteniendo la estabilidad y potencia del lenguaje C++. Es importante entender esta movilidad o flexibilidad de niveles y lenguajes.



GAmuza está programado para funcionar en mac OSX y Ubuntu Linux x86 de 32 y 64 bits. Como se ha señalado, hay distintas versiones de GAmuza que tienen requisitos de sistema, procesos de instalación e interfaces diferentes, en este texto vamos a hacer referencia a la versión 043x, para otras versiones ver nota al pie[13].

Para **instalar** GAmuza 043x, desde <http://gamuza.d3cod3.org> descargad la aplicación, abrir (montar) el archivo .dmg y arrastrar el icono de GAmuza a la carpeta Aplicaciones. El proceso de instalación ha terminado.

13 Para las versiones 0420 OSX 10.6 y 0399 Ubuntu ver el archivo http://mpison.webs.upv.es/mie1/text/GAmuza_0420_manual.pdf

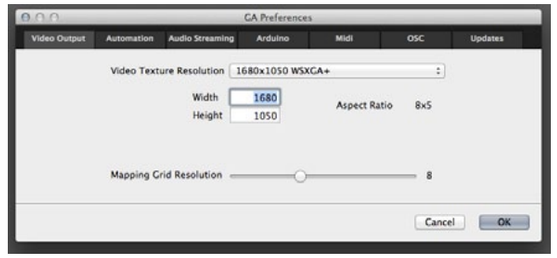
1.1. Configuración Preferencias

GAmuza 043x tiene el ajuste de configuración del proyecto en el menú **Preferences**. Para activarlo, clicar sobre el nombre del programa situado en la esquina superior izquierda de la pantalla, se despliega un menú cuyo segundo ítem es **Preferences** (también se puede activar directamente tecleando Comando + coma ⌘,)

La ventana **GA Preferences** tiene 7 pestañas para configurar.

Video Output (Ventana de salida):

Video Texture Resolution. Es un menú desplegable para seleccionar la resolución de la ventana de salida, se debe elegir según la resolución de los monitores o proyectores que se utilicen como segunda pantalla; las últimas opciones hacen referencia al uso de tarjetas que duplican o triplican el tamaño de la ventana de salida.

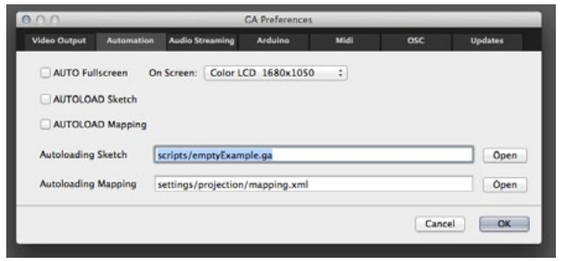


El segundo ítem está relacionado con el anterior y permite poner manualmente la resolución.

El tercer ítem es **Mapping Gird Resolution**, GAmuza genera una rejilla sobre la imagen de salida para proyecciones de mapping. Con este slider se puede elegir el número de nodos de esa rejilla, desde un mínimo de 1 (significa que el rectángulo de la proyección tiene un punto de ajuste activo en cada vértice, con los que se puede deformar la proyección según las necesidades), a un máximo de 20 (la imagen tiene 400 rectángulos ajustables). En la descripción de la Barra de Menús se señalan más datos sobre el funcionamiento de esta rejilla.

Automation:

Opción pensada para la presentación de instalaciones. Permite seleccionar un archivo de script previamente guardado, "**Autoloading Sketch**", abriendo una ventana de diálogo para elegirlo dentro del ordenador. Y también si hay un archivo guardado con una configuración de



mapping con "**Autoloading Mapping**". Si se activa **AUTO fullscreen**, al abrirse el programa, la ventana de salida se pone directamente a pantalla completa, y si se activa **AUTOLOAD Scketch**, GAmuza se inicia procesando el archivo de script seleccionado. Igualmente, si se activa **AUTOLOAD Mapping**, la imagen mostrará las deformaciones que contenga el archivo de configuración de mapping seleccionado.

Normalmente las exposiciones duran varios días y mantener técnicamente las piezas de arte interactivo puede ser un problema, si se conjuga la configuración de **Automation** con las opciones que hay en las Preferencias del Sistema del ordenador para arrancar y apagar el equipo en determinados días y horas, y que GAmuza se abra automáticamente al iniciar la sesión, la pieza requerirá un mantenimiento mínimo durante la exposición.

Audio Streaming

Input Device muestra un menú desplegable con los dispositivos de entrada de audio que reconoce el ordenador, hay que seleccionar con cuál trabajará GAmuza.

Output Device, semejante al anterior, hay que seleccionar el dispositivo de salida de audio.

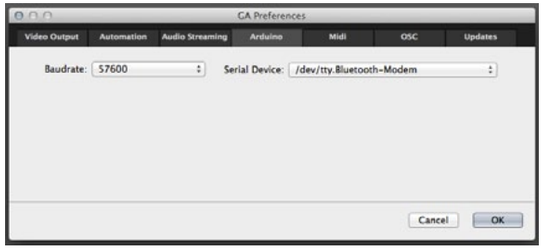
En la opción **Sampling Rate** (Frecuencia de muestreo) hay que seleccionar un número que sea acorde con el dispositivo de entrada de audio. Se puede ver las frecuencias que soporta ese dispositivo en la ventana **Tools/Logger** que se describe en el capítulo siguiente.

Buffer Size, selecciona el tamaño de Buffer con que se va a trabajar (cantidad de muestras almacenadas), 512 es un tamaño válido para la mayoría de los proyectos, porque no produce clics o chasquidos en el audio, pero si se va a trabajar con Bins o FFT (ver pág. 191) tal vez requiera un tamaño mayor, hay que tener en cuenta que a mayor buffer más recursos de procesador consume.

FFT Windowing (Ventanas de función FFT). Las ventanas son funciones matemáticas que se usan para el análisis y procesamiento de la señal de audio porque evitan las discontinuidades al principio y final de los bloques analizados. En el menú desplegable se puede seleccionar entre: Rectangular, Bartlett, Hann, Haming o Sine.[14]

14 Para conocer las características de estas opciones ver < http://es.wikipedia.org/wiki/Ventana_(funcion) > [25.07.2013]

Arduino

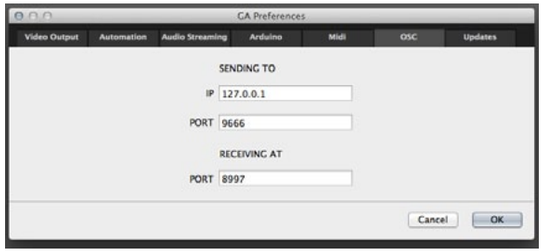


Baudrate (57600 por defecto) y **Serial Port** (puerto serie). Puede utilizarse el software de Arduino para comprobar mediante el Serial Monitor cuál es el puerto asignado y seleccionar en GAmuza el mismo.

Midi

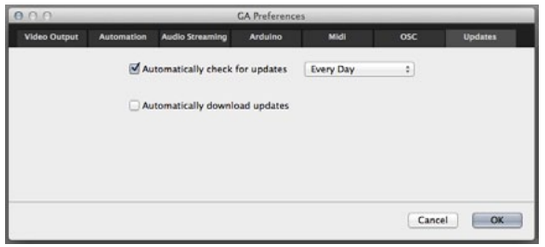
Aunque no existe un módulo GUI de MIDI, GAmuza puede recoger datos de estos dispositivos y hay funciones específicas para comunicar con ellos.

OSC



Contiene los ajustes para el protocolo de transferencia de datos OSC, (Open Sound Control)[15]. Los datos se introducen directamente en los campos de texto. Losdos primeros SENDING TO IP y PORT indican el número de IP y puerto donde se enviarán eventuales datos. El último, **RECEIVING AT PORT** indica el puerto de este ordenador que recibirá los datos enviados desde otro.

Updates



Para configurar el modo en que queremos actualizar el software.

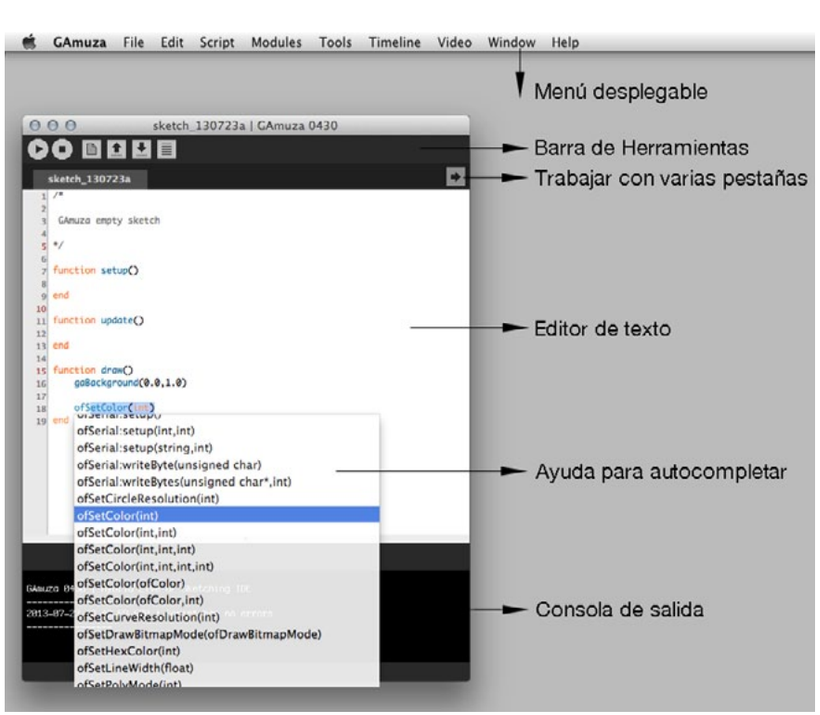
15 Ver definición de OSC en http://es.wikipedia.org/wiki/OpenSound_Control [12.05.2012]

2. Descripción de la interface

2.1. IDE de programación

La IDE (integrated development environment) es un entorno de programación empaquetado como aplicación para facilitar la escritura del código. Al abrir la aplicación GAmuza.app, tras la ventana de créditos, aparece un archivo de script en el IDE, con un modelo de bloques de programación básico preconfigurado, y la ventana de salida con las dimensiones que tenga asignadas en **Preferences.**

La interface del IDE de programación sigue directamente el modelo marcado por el PDE Processing, como epígono de una de sus influencias directas. Principalmente aporta un editor de texto para el código que en su parte superior dispone una barra de herramientas y en su parte inferior una consola de salida. Tiene también un sistema de ayuda para autocompletar el nombre de las funciones (Code Completion) que se activa presionando la tecla Esc después de iniciar el nombre de la función, en el caso de los métodos para las clases de openFramworks, hay que escribir el nombre de la clase, después teclear Esc, y aparecen todos los métodos de esa clase.



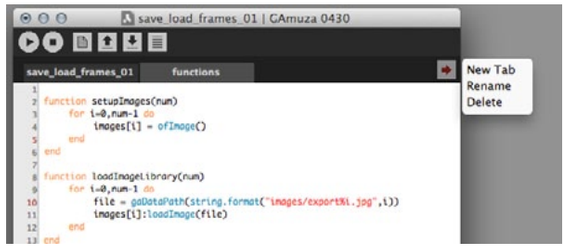
En la barra de herramientas hay botones que facilitan las labores más usuales: enviar a GAmuza, limpiar script, nuevo script, abrir script, guardar como, y limpiar la consola de salida.

El editor de texto tiene una columna numerada a la izquierda para facilitar la localización de errores. También asigna una serie de colores a términos que reconoce, vinculados a:

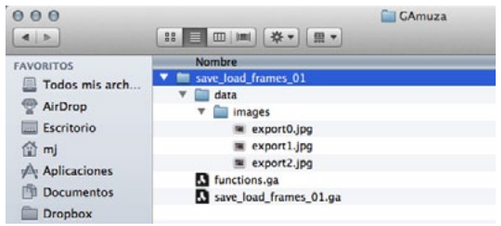
- Funciones, clases, métodos y constantes: azul — `ofSetColor(255)`
- Estructuras programación: naranja — `function`
- openGL, funciones: azul turquesa, y constantes: marrón pardo — `glBegin(GL_QUADS)`
- Strings (cadena caracteres entrecomillados): rojo — `"texto entrecomillado"`
- Comentarios: gris — `// comentario`
- Negro para todo lo demás. — `x = 125`

Al guardar el archivo por primera vez, Gamuza genera una carpeta con el mismo nombre que se le dé al script y guarda junto a él otra carpeta denominada **data**, en esta carpeta se deben poner todos los medios que utilicemos: fuentes de texto, videos, audio, imágenes...

El botón con el icono de una flecha, situado en la parte superior derecha del IDE, sirve para incorporar, renombrar o borrar, archivos de script vinculados al principal. Estos archivos se presentan como varias pestañas en el IDE y son guardados en su misma carpeta.

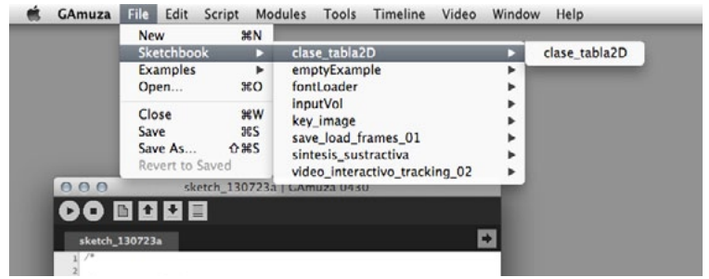


La siguiente imagen muestra la relación de archivos y carpetas de un script con dos pestañas, donde se ha incorporado manualmente otra carpeta dentro de data para almacenar las imágenes que se exportan.

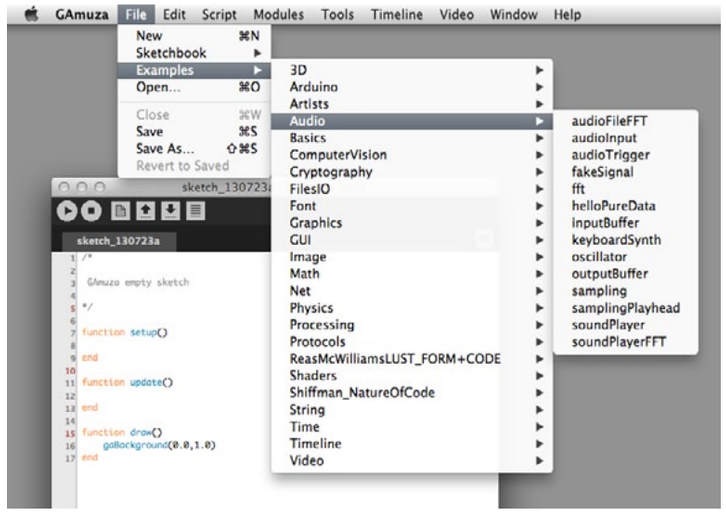


Los dos archivos de script aparecen con la extensión .ga; si el archivo principal se vuelve a guardar, se actualiza directamente en esa carpeta, pero si se da a "guardar como" (desde el botón de la barra de herramientas o el menú superior desplegable), y se aloja en otro sitio, hay que volver a copiar los elementos que se utilicen en la nueva carpeta **data** que genere.

2.2. Barra de Menú



En el primer menú, **File**, se accede a las operaciones habituales para abrir un archivo de script nuevo (**New**) o que ha sido previamente guardado (**Open**). Entre ambas opciones se encuentran dos ítems que tienen un carácter diferente; el primero, **Sketchbook**, da acceso directamente a la carpeta GAmuza situada en Documentos, donde se recomienda guardar los archivos de script. El segundo, **Examples**, contiene numerosos ejemplos que vienen con el programa, ordenados por categorías. Todos están comentados mostrando el uso de las funciones propias de GAmuza y de una amplia tipología de prácticas.



Los tres siguientes ítems del menú **File** corresponden a las tareas usuales de cerrar (**Close**), guardar (**Save**) y guardar como (**Save As**). El último, recupera la versión anterior guardada de ese script (**Revert to Saved**). Junto a los ítems del menú se muestra el atajo de teclado para poder hacer esa tarea sin tener que acceder al menú.

El contenido del menú desplegable **Edit** es el similar al de la mayoría de los software: **Undo** (⌘Z, deshacer), **Redo** (⇧⌘Z, rehacer), **Cut** (⌘X, cortar), **Copy** (⌘C ,copiar), **Paste** (⌘V, pegar), **Select All** (⌘A seleccionar todo), **Find** (⌘F, encontrar y reemplazar), y por último **Special Characters** (caracteres especiales) que abre el Visor de caracteres de Mac.

En el menú **Script**, la opción **Compile Script** (⌘K) equivale al botón play (Send to GAmuza) de la barra de herramientas del IDE, y **Clear Script** al segundo botón, stop. El tercer ítem **Clear Console** equivale al último de los botones y borra el contenido de la consola de salida.

El primer ítem del menú **Modules** es Preview (⇧⌘0) y activa/desactiva la ventana del previo de imagen. Los siguientes ítems corresponden a los módulos GUI del TimeLine, Análisis de audio y Arduino que se analizará con detalle en los capítulos 8, 9 y 10.

El menú Tools da acceso a las herramientas de GAmuza:

Color Correction (⇧⌘K) cuenta con tres bloques de efectos preprogramados para ajustar, mediante sliders, la apariencia de color de la imagen en la ventana de salida:

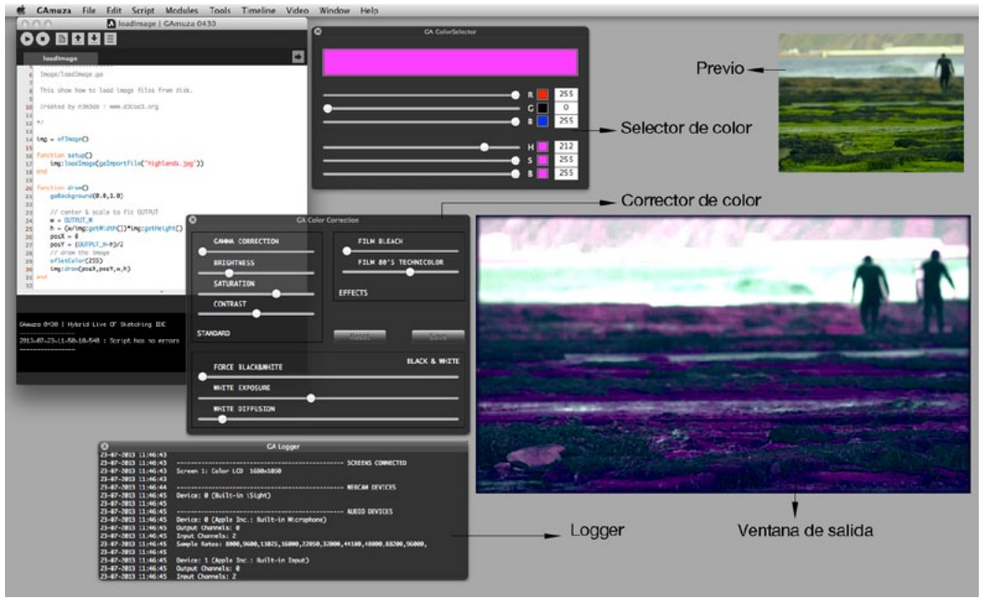
- **Gamma correction**: ajusta la "sensación" de contraste de la imagen si se percibe blanquecina o "lavada".
- **Brightness**: ajusta el brillo de la imagen.
- **Saturation**: satura o desatura el color de la imagen.
- **Contrast**: ajusta el contraste.
- **Film Bleach**: emula una técnica de revelado analógico que produce contraste alto y saturación baja. También llamado Bleach Bypass.
- **Film 80's Technicolor**: emula el rango de color de los TV de tubo catódico de los años 80's.
- **Force Black&White**: Estos tres últimos filtros están relacionados y el ajuste de uno repercute en los otros. Este aporta otra manera más flexible para desaturar.
- **White Exposure**: ajusta la sensación de luz.
- **White Diffusion**: emula el uso de filtros de gelatina White Diffusion para suavizar sombras.

Color Selector (⇧⌘C), ayuda a conocer los parámetros de un determinado color en los modelos RGB y HSB.

Logger, es una ventana que muestra todos los dispositivos que reconoce el ordenador: pantallas, cámaras, tarjetas de sonido, puertos serial, Arduino, conexión OSC, así como si el Pure Data Synthesis Engine está iniciado, el listado de plugins de audio (Audio Unit) disponibles en el sistema, y si la tarjeta gráfica del equipo soporta el uso de shaders.

Archive sketch, crea un archivo comprimido .zip con el script y todos sus contenidos, para guardarlo en el ordenador o enviar por correo electrónico.

Por último, **Export to HTML**, recoge el script con todos los ficheros utilizados, un par de pantallazos de la ventana de salida, una copia del setting de los módulos, y además crea un archivo .html con una página de documentación del script automáticamente maquetada, actuando como un sistema de documentación automática del trabajo.



Los ítems que contiene el menú **Timeline** son los habituales cortar, copiar, pegar, pero el atajo de teclado varía, en lugar de la tecla comando ⌘, utiliza Control ^. Su uso se describirá en el capítulo dedicado al Timeline.

En el menú **Video** aparecen opciones para:

Mapping. **Edit Mapping**, para mostrar/ocultar la rejilla de mapping sobre la ventana de salida

Reset Mapping, Deja la rejilla ortogonal eliminando todas las deformaciones.

Keyboard Control (⇧⌘M), pasa el control de ajuste de los nodos de la rejilla del ratón al teclado y viceversa.

Manual Edit Point ON (⇧⌘↩) Activa un nodo para poder moverlo.

Manual Edit Point OFF (⇧⌘⌫) Desactiva el nodo para poder pasar a otro.

Go to North Point (⌘↑) Si el nodo está activo, lo desplaza hacia arriba, si se ha desactivado pasa el control al nodo superior.

Go to South Point (⌘↓) Si el nodo está activo, lo desplaza hacia abajo, si se ha desactivado pasa el control al nodo inferior.

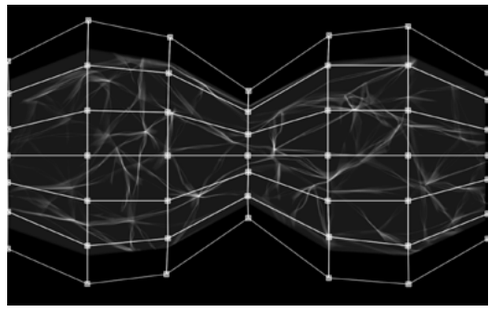
Go to East Point (⌘→) Si el nodo está activo, lo desplaza hacia la derecha, si se ha desactivado pasa el control al nodo de la derecha.

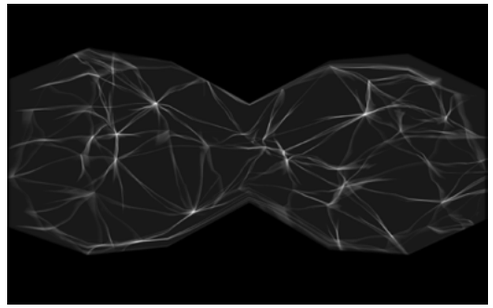
Go to West Point (⌘←) Si el nodo está activo, lo desplaza hacia la izquierda, si se ha desactivado pasa el control al nodo de la izquierda.

Load Mapping. Para cargar una configuración de mapping previamente guardada.

Save Mapping. Para guardar la configuración actual.

Los ajustes de mapping se realizan proyectando sobre la superficie irregular que se va a mapear por lo que la ventana de salida está en el proyector, y la imagen repta por esa superficie pudiendo quedar algunas zonas con poca visibilidad desde el punto de vista en que se encuentre el ordenador, para estos casos es importante tener el control de los nodos con el teclado, lo que permitirá ver cómo se ajusta la deformación de la imagen a las superficies aunque el nodo del que depende esa zona esté en un lugar al que no podemos llegar visualmente para clicar con el ratón.





Preview. Para seleccionar el tamaño de la ventana del previo. Las opciones son: QVGA (320 x 240), nHD (640 x 360), VGA (640 x 480), SVGA (800 x 600) y FWVGA (854 x 480)

Toggle Full Screen (⌘F) Para activar/desactivar la ventana de salida a pantalla completa.

3. Interface Módulo Timeline

Como vimos en la descripción del Menú desplegable [ver página 25], GAmuza incorpora un módulo con interface gráfica para Timeline que facilita la programación al poder visualizar el tiempo y poner keyframes (fotogramas-clave) para trabajar con color y curvas de transformación que interpolan sus valores entre los keyframes, así como utilizar flags, bangs o switches para activar cambios a lo largo del Timeline[16].

Los datos de estos cambios se almacenan en archivos de configuración .xml que se van almacenando en la carpeta data de script. Por eso la primera función para generar un Timeline es:

```
gaTimelineSetup(string, string)
```

Su primer parámetro indica el directorio donde se guardaran esos archivos de configuración, y para que se sitúen en la carpeta data se utiliza la función de GAmuza `gaDataPath(string)`, siendo ese string el nombre de la carpeta donde se van a guardar los archivos del timeline, en el caso de que no sea la propia carpeta data ha de generarse manualmente dentro de ella. El segundo parámetro es el nombre que tendrá el timeline para esos archivos. En el ejemplo que viene a continuación estos datos son:

```
gaTimelineSetup(gaDataPath("timeline/"),"timelineTest")
```

Vamos a ir analizando la incorporación de pistas al timeline y los archivos que genera, a través del siguiente ejemplo:

```
/*
 GAmuza 043
 ---------------------
 Timeline/basicTimeline.ga
 presionar barra espacio para reproducir/parar el timeline
 created by n3m3da | www.d3cod3.org
*/

circleX = 0
circleY = 0

function setup()
   ofSetCircleResolution(50)

   gaTimelineSetup(gaDataPath("timeline/"),"timelineTest")
   gaTimelineSetLoopType(OF_LOOP_NORMAL)
   gaTimelineSetFrameRate(25)
   gaTimelineDurationInSeconds(10)
```

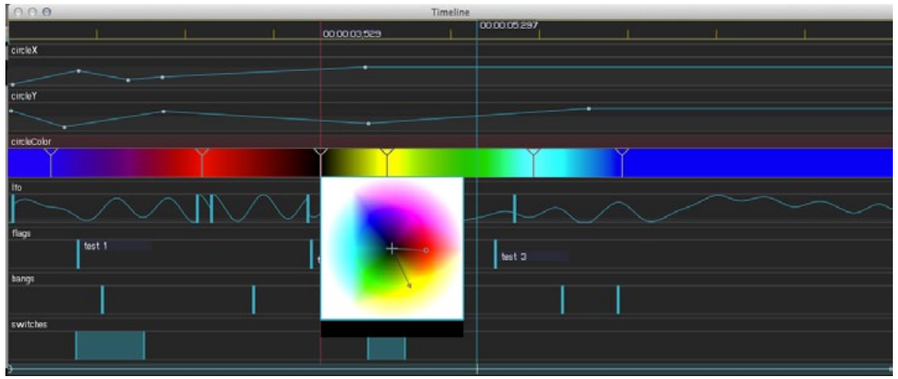
16 La base del TimeLine procede de James George, ofxTimeline <http://jamesgeorge.org/opensource.html> original source code: <https://github.com/YCAMInterlab/ofxTimeline> [12.08.2013]

```
    gaTimelineAddCurves("circleX",0,OUTPUT_W)
    gaTimelineAddCurves("circleY",0,OUTPUT_H)
    gaTimelineAddColors("circleColor")
    gaTimelineAddLFO("lfo")
    gaTimelineAddFlags("flags")
    gaTimelineAddBangs("bangs")
    gaTimelineAddSwitches("switches")
    gaTimelinePlay()
end

function update()
    circleX = gaTimelineGetValue("circleX")
    circleY = gaTimelineGetValue("circleY")
end

function draw()
    gaBackground(1.0,0.03)

    ofSetColor(gaTimelineGetColor("circleColor"))
    ofNoFill()
    ofCircle(circleX, circleY,gaTimelineGetValue("lfo")*300)

    ofSetColor(0)
    ofDrawBitmapString(tostring(gaTimelineGetBang()),200,200)
    ofDrawBitmapString(tostring(gaTimelineGetSwitch("switches")), 200, 250)
end
```

Las primeras cuatro líneas de código vinculadas al timeline en el bloque `setup()`, configuran la base del timeline: la ruta de los archivos, el tipo de loop que sigue la reproducción del Timeline, el frame rate y su duración en segundos. Después, `gaTimelineAddCurves("circleX",0,OUTPUT_W)` genera un track (o pista) llamado **`circleX`** que, verticalmente, en su parte inferior pasa al sistema el valor 0 y en su parte superior el valor `OUTPUT_W`, interpolando entre uno y otro los valores intermedios. Lo mismo para el track **`circleY`**.

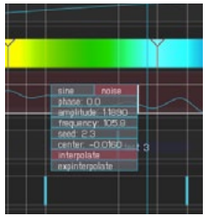
Si después de poner estas líneas de código abrimos la carpeta data/timeline veremos que se han generado los archivos **`timeline0_Page_One_trackPositions.xml`** y **`timeline0_timelineTest_trackPositions.xml`**, en este último se irán almacenado los datos de todos los tracks, y al abrir el Módulo del Timeline, en cuanto empecemos a clicar keyframes en esos tracks se generan los archivos **`timeline0_circleX.xml`** y **`timeline0_circleY.xml`**, de manera que el código genera el track y al señalar keyframes en esos tracks se van creando archivos .xml que guardan su configuración, lo mismo para la pista de color, los flags, bangs, switches y lfos, un archivo para cada uno de ellos.

Al clickar en la pista **`circleColor`**, se sitúa un keyframe y aparece una representación del sistema de color, al clicar sobre un determinado color queda asignado ese valor para ese keyframe. Si se establece otro keyframe con otro color, entre ambos se despliega una escala cromática entre ambos tonos.



Para recoger estos valores de color se utiliza la función `gaTimelineGetColor("circleColor")`, en el ejemplo es el parámetro de `ofSetColor()` para los círculos.

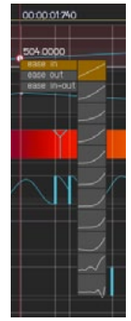
La función `gaTimelineAddLFO("lfo")` genera un tipo de pista que simula una oscilación de baja frecuencia (Low Frequency Oscillation), al poner en ella un keyframe y clicar sobre él con el botón derecho aparece una tabla de configuración, se puede seleccionar curva tipo seno o noise y para las características de la curva que se regulan arrastrando con el ratón a derecha o izquierda de cada uno de los valores.



En el ejemplo, los valores de estas curvas se recogen con la función `gaTimelineGetValue("lfo")` para aplicarlos al diámetro del círculo.

La función `gaTimelineAddFlags("flags")` genera una pista cuyos keyframes pueden llevar etiquetas, mientras que la función `gaTimelineAddBangs("bangs")` genera un pista en la que cada keyframe señala un bang que puede actuar como disparador para inicializar eventos. Tanto los flags como los bangs pueden leerse con la función `gaTimelineGetBang()` , con la diferencia de que en los primeros podemos diferenciar el string de la etiqueta concreta cuando la línea de reproducción del Timeline alcanza su posición.

La función `gaTimelineAddSwitches("switches")` genera una pista donde situar zonas más o menos amplias de interruptores que devuelven al sistema un valor boolenano, `true` o `false`.



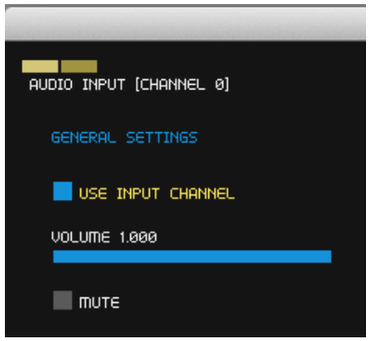
Por último señalar que la interpolación entre los valores de los keyframe de las primeras pistas generadas con la función `gaTimelineAddCurves()` pueden seguir diferentes tipos de curva. Estas opciones aparecen al clicar con el botón derecho sobre el keyframe.

4. Interface Módulo Audio Streaming

Antes de describir las opciones de la interface, señalar que este módulo GUI es solo para ajustar el análisis de entrada de sonido (Input Channel), no la salida. Si el dispositivo seleccionado tiene varios canales de entrada, veremos varios interfaces de Audio Input iguales que permiten ajustes independientes, a cada uno de ellos se accede clicando en una de las pestañas amarillas. Debajo de las pestañas se indica el canal que estamos visualizando [CHANNEL 0], [1], [2]..., que será necesario poner como parámetro en muchas de las funciones de programación de audio.

Si sólo se va a analizar la entrada de un canal, se pueden desactivar los otros con el botón Use Input Channel.

Debajo hay un slider para regular el nivel de volumen y un botón para silenciarlo.



OSC Data Settings. El slider Smoothing factor suaviza la relación de los datos a enviar por OSC cuando presentan grandes cambios, utilizando un filtro de Kalman.

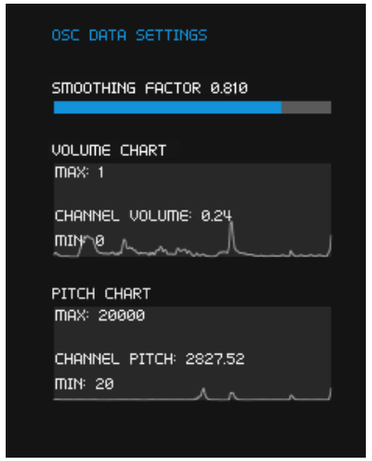
Volume Chart muestra el gráfico y valores del volumen. Su rango de valores va de 0.0 a 1.0. La visualización de estos datos permite calcular mejor si deben escalarse, y por cuánto, para su uso en programación. La función de GAmuza que recoge el volumen es `gaGetVolume()`.

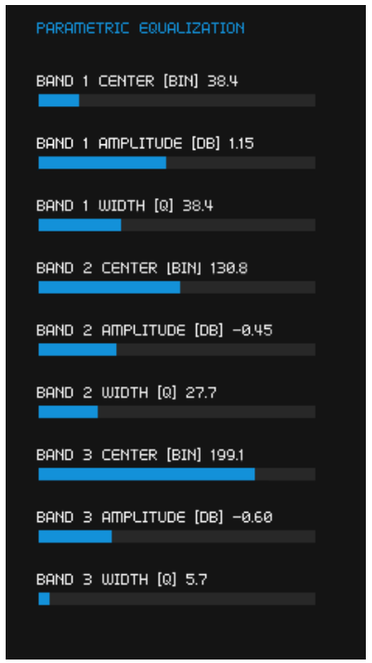
Pitch Chart muestra igualmente el gráfico y valores del tono (pitch), su rango oscila entre 20Hz to 20000Hz, pero cuando los recoge la función de GAmuza, `gaGetPitch()`, los normaliza entre 0.0 y 1.0

El siguiente bloque de ajustes, Parametric Equalization, prepara la señal para la transmisión de datos, ajustando el control individual de tres parámetros, en tres bandas: frecuencia central, ganancia y ancho de banda.

- Center [BIN], selecciona la frecuencia central en Hercios. Dependiendo del valor que se ha asignado al BufferSize en el panel de preferencias, el número de BINs que tendrá la FFT será de (BufferSize/2) + 1. Por ejemplo con un BufferSize de 512 se realizará una analisis FFT sobre 257 BINs, es decir 257 grupos de frecuencias equidistribuidas entre 20Hz y 20000Hz.
- Amplitude [DB], controla la ganancia, determinando cuánto se amplifican o recortan esas frecuencias, por eso puede tener valores negativos, como muestra la imagen y el gráfico en la segunda banda.
- Width [Q], el factor Q determina la nitidez o calidad de la anchura de banda.

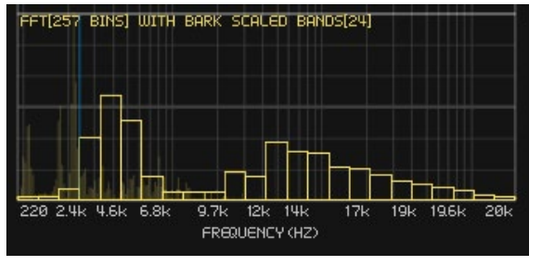
Los ajustes se van visualizando en el gráfico Parametric EQ. Primero se busca la posición de la frecuencia central y se le ajustan la ganancia y el factor Q.





Hay otros dos datos que GAmuza obtiene del análisis FFT: **Bark Scaled Bands** y **Standard FFT**.

La escala de Bark[17], es una escala psicoacústica propuesta por Eberhard Zwicker en 1961, que corresponde a la subdivisión de las primeras 24 bandas críticas del oído. Los márgenes de las bandas en Hercios son 0, 100, 200, 300, 400, 510, 630, 770, 920, 1080, 1270, 1480, 1720, 2000, 2320, 2700, 3150, 3700, 4400, 5300, 6400, 7700, 9500, 12000, 15500.



GAmuza obtiene el valor de la escala de Bark desde el análisis del FFT. Para trabajar con estos datos se utiliza la función `gaGetBin(int,int)` y la variable interna `FFT_BANDS`.

La transformada rápida de Fourier, FFT (Fast Fourier Transform), es un algoritmo muy utilizado para el tratamiento de la señal de audio. En GAmuza opera sobre las muestras de sonido almacenadas en el BufferSize, por lo que su rango de valores depende del tamaño de buffer que se haya asignado en **Preferences**.

Si hay un tamaño de buffer 512, el gráfico FFT tendrá un tamaño [257 BINS], si el buffer es 1024, ese valor cambia a [513], por lo que el algoritmo establece que el tamaño del FFT es el del BufferSize /2 +1, que serán los puntos de análisis de la frecuencia.

En programación, estos datos se utilizan con la función `gaGetFFT(int,int)` que tiene dos parámetros: ID del canal de entrada de audio:`[CHANNEL num]`, y la posición que ocupa la muestra (sample) en el Buffer, recorriéndolo desde 0 al valor de la variable global interna `BUFFER_SIZE`.

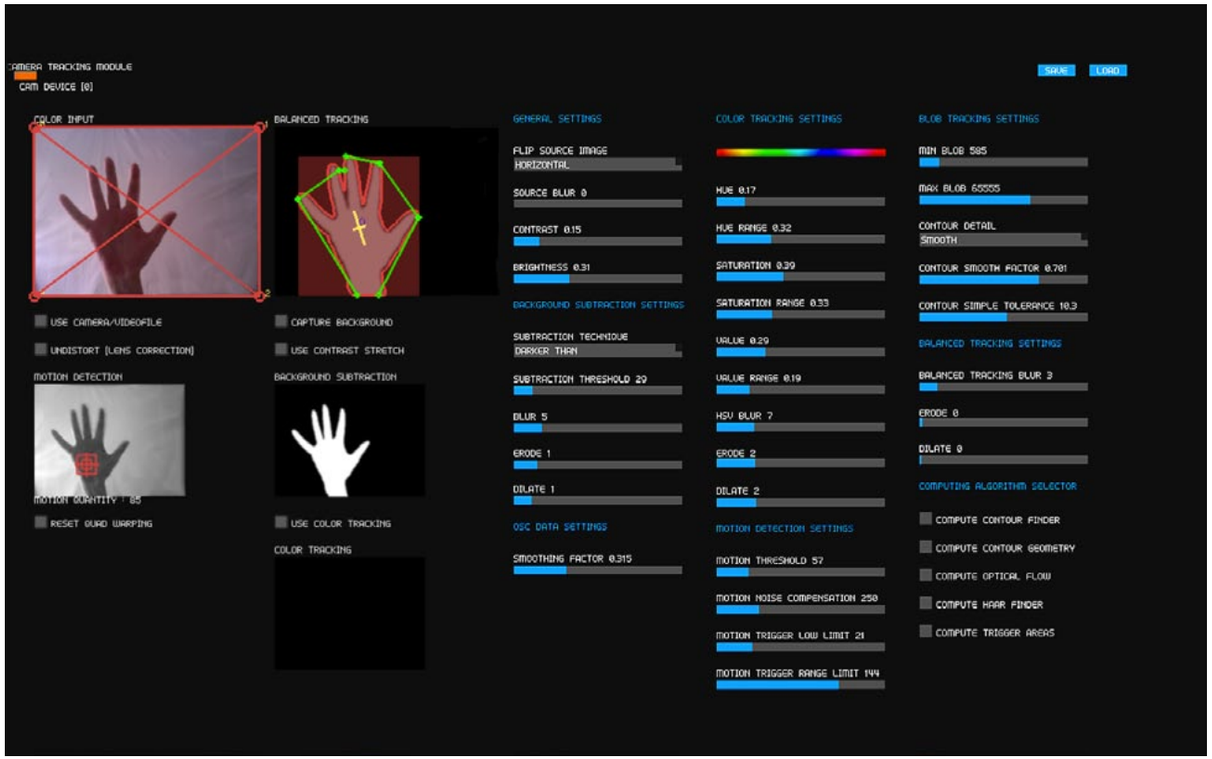
17 Definición Bark scale [enciclopedia on-line] http://en.wikipedia.org/wiki/Bark_scale [25.08.2012]

5. Interface Panel Tracking Video (Cameras)

En GAmuza se pueden hacer scripts para tracking video utilizando las clases y métodos propios de opneFrameworks `ofVideoGrabber()` u `ofxKinect()` para detectar la fuente y los addons de openCV `ofxCvColorImage()`, `ofxCvContourFinder()`, `ofxCvFloatImage()`, `ofxCvGrayscaleImage()`, `ofxCvShortImage()`, `ofxCvHaarFinder()`, `ofxOpticalFlowLK()`, o utilizar el panel de Computer visión con interface GUI que ha sido preconfigurado principalmente para facilitar este proceso a aquellos que no sean expertos en la programación.

A diferencia de los módulos, los paneles se generan por programación en la ventana de salida usando las clases de GAmuza `gaCameraTracking()` o `gaKinectTracking()`, se pueden activar desde los ejemplos ComputerVision/camTrackingPanel para cámaras de video o ComputerVision/kinectTrackingPanel cuando se trabaja con la kinect.

En este capítulo se describe la interface del panel de Tracking para cámaras de video y los métodos de la clase de GAmuza `gaCameraTracking()`, explicándolo a partir del código que genera su carga.



```
/*
 GAmuza 043
 ---------------------
 Panel de tracking video de GAmuza
 created by n3m3da | www.d3cod3.org
*/

camPanel = gaCameraTracking()
drawGUI = true
camID = 0

captureWidth = 320
captureHeight = 240
cam = ofTexture()

function setup()
   camPanel:setGuiSettingsFile(gaDataPath("camTrackingSettings.xml"))
   camPanel:setup(camID,captureWidth,captureHeight)

   cam:allocate(captureWidth,captureHeight,GL_RGB)
end

function update()
   cam = camPanel:getCameraTextureMod()
   camPanel:update()
end

function draw()
   gaBackground(0.1,1.0)
      // para mostrar la imagen en direrecto de la cámara
   ofSetColor(255)
   scaleH = OUTPUT_H
   scaleW = scaleH* captureWidth / captureHeight
      cam:draw(OUTPUT_W/2 - scaleW/2,0, scaleW,scaleH)

      ofSetColor(255)
      if drawGUI then
         camPanel:draw()
      end
end

function keyReleased()
   if gaKey() == string.byte('g') then
      drawGUI = not drawGUI
   end
end
```



```
function mouseDragged()
   camPanel:mouseDragged(gaMouseX(),gaMouseY())
end

function mousePressed()
   camPanel:mousePressed(gaMouseX(),gaMouseY())
end

function mouseReleased()
   camPanel:mouseReleased(gaMouseX(),gaMouseY())
end
```

Como en otros trabajos con clases, inicialmente se asocia una variable global a la clase `gaCameraTracking()`; la segunda variable es booleana y permitirá visualizar/ocultar el panel en la ventana de salida; las dos siguientes definen las dimensiones del video capturado, y la última variable está vinculada a la clase `ofTexture()` como soporte de la imagen capturada por la cámara de video

En el bloque `setup()` se inicializa el panel con el método `setGuiSettingsFile` cuyo parámetro indica la ruta y nombre del archivo xml de que guardará la configuración del panel. Al activar el script aparece el panel en la ventana de salida, tras realizar los ajustes que requiera el proyecto, debe guardarse esa configuración clicando el botón "save" situado en el vértice superior derecho del panel. Al guardarlo se genera el archivo `camTrackingSettings.xml` en la carpeta data del script, si no se guarda el script no funcionará. El método `setup` tiene como parámetros el ID de la cámara y el tamaño del frame. Después con el método `allocate` de la clase `ofTexture()` se asignan el tamaño y tipo de color GL del frame de vídeo.

En el bloque `update()`, se identifica la textura con el panel a través del método `getCameraTextureMod()` [devuelve la captura de la camara con el eventual mirror aplicado en horizontal/vertical] o con `getCameraTexture()` [devuelve la captura desde la camara en "raw"], con cualquiera de ellos se relaciona la clase de openFrameworks con la de GAmuza, y después con el método `update` se actualiza la información de la imagen capturada cada frame.

En el bloque `draw()`, se dibuja la imagen en directo de la cámara adaptada a pantalla completa, y después, para que quede encima de la imagen, se dibuja el panel cuando la variable booleana `drawGUI` es verdadera.

En el bloque `keyReleased()`, se establece una condicional para que al teclear la letra "g" la variable `drawGUI` pase de verdadera a falsa y viceversa, construyendo un switch para ver u ocultar el panel.

Los bloques `mouseDragged()`,`mousePressed()` y `mouseReleased()` permiten manipular con el ratón los sliders y botones de la interface del panel

5.1. Descripción panel

El panel Tracking video permite analizar características visuales del campo que registra la cámara, recoger determinados datos como la luminosidad, color o movimiento, seleccionar y/o ajustar esos datos, para después generar con ellos otra situación que responda en tiempo real a sus cambios, facilitando la aplicación de los algoritmos para tracking video al regular con sliders, menús y botones las condiciones básicas de entrada de datos al sistema.

GAmuza recoge el reconocimiento que el ordenador ha hecho de la cámara o cámaras conectadas, asignando un ID a cada una de ellas, empezando por el 0, es decir, si solo hay una cámara conectada su id es 0, si hay dos la primera es 0 y la segunda 1 y así sucesivamente.

Las interface del panel está distribuida en función de las distintas técnicas básicas para tracking video:

Blob detection, reconocimiento de regiones o áreas. Puede detectar los blobs por **Background substraction** o por **Color tracking.** Para el reconocimiento del contorno debe activarse el algoritmo **Compute Contour Finder** y para la geometría del contorno hay que añadir el algoritmo **Compute Contour geometry**

Motion Detection, reconoce la cantidad de movimiento y localiza el punto medio de la masa, siempre está activo aunque ningún algoritmo haya sido seleccionado

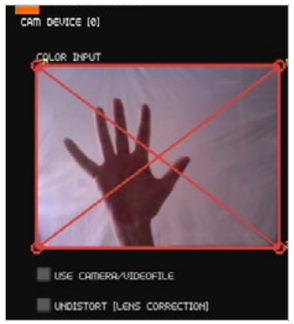
Haar Finder, para reconocimiento de partes del cuerpo humano, el algoritmo **Compute Haar Finder** debe estar seleccionado

Optical Flow, dirección del movimiento representado vectorialmente, debe estar seleccionado el algoritmo **Compute Optical Flow**

Trigger Areas, son nueve áreas de activación ajustables en tamaño y posición desde el módulo **Computer vision**. Debe activarse el algoritmo **Compute Trigger Areas**

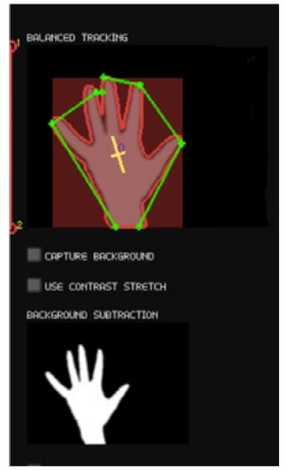
Análisis detallado de la interface

El monitor de entrada de la señal video, **Color input**, permite reducir el área a trackear, este *crop* se ajusta arrastrando con el ratón los puntos resaltados de los vértices. Los demás monitores muestran solo la zona seleccionada. El crop se resetea con **Reset Quad Warping.**



USE CAMERA/VIDEOFILE. Activa/desactiva la posibilidad de trabajar con un archivo de video pregrabado (videofile) o con la imagen capturada por la cámara en directo.

El monitor **Balanced Tracking** muestra el resultado de todos los algoritmos de tracking seleccionados y ajustados con los sliders, menús y botones del bloque derecho.



Capture Background, al clicar el botón se captura un frame de la imagen video que debe corresponder con el fondo. Esta imagen se toma como referencia para comparar los cambios que se producen entre el valor de sus píxeles y el de los frames de la imagen a trackear, aplicando el algoritmo Background Subtraction. Es el primer nivel para el cálculo de Tracking, y siempre está activo. Según las condiciones de iluminación, puede mejorarse la imagen activando **Use Contrast Stretch**: incrementa el rango de los valores más intensos de la imagen en escala de grises.

El resultado de **Background substraction** se ve en el monitor inferior.

GENERAL SETTINGS

Flip Source Image voltea la imagen-video que proviene de la cámara, su ajuste depende de las condiciones de exhibición:

- **Off**: la imagen se percibe sin ningún cambio
- **Horizontal**: se selecciona cuando los movimientos ante la imagen responden al efecto de estar frente a un espejo.
- **Vertical**: imagen video cabeza abajo.
- **Horizontal + Vertical**: voltea en los dos sentidos.

Source Blur: desenfoque de la imagen de entrada de video.
Contrast: contraste. **Brightness**: brillo.

BACKGROUND SUBTRACTION SETTINGS

Substraction Technique, seleccionar entre:

- **Color ABS**: color absoluto
- **B&W ABS**: blanco y negro absoluto
- **Lighter than**. Cuando la figura a trackear es más clara que el fondo.
- **Darken than**. Cuando La figura a trackear es más oscura que el fondo.

Substraction Threshold, ajusta el umbral de detección. Cuanto menor es el valor más sensible. Si se trabaja con Tracking Color se pone el valor máximo para anular los datos de substracción del fondo

Blur, desenfoca la imagen procesada. **Erode**, reduce las superficies detectadas. **Dilate**, expande las superficies detectadas

OSC DATA SETTINGS

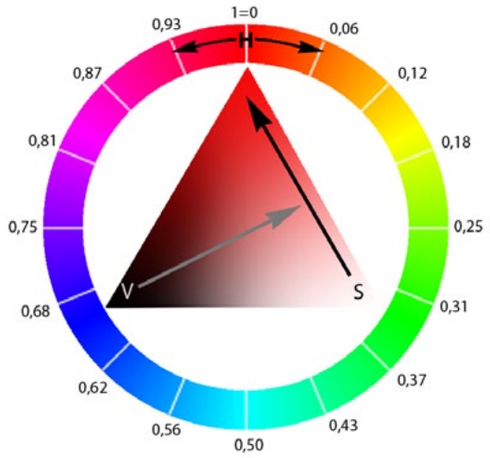
Smoothing factor ajusta el rango de interpolación.

USE COLOR TRACKING

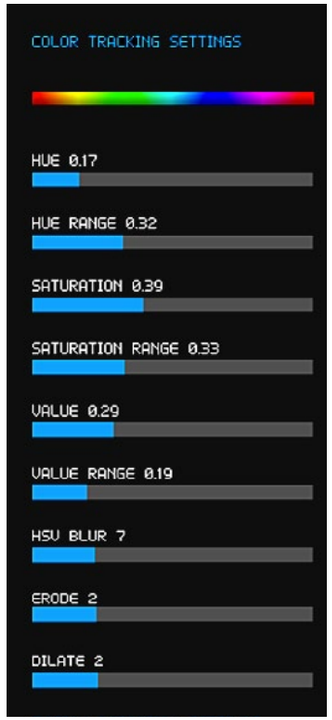
Color Tracking es un algoritmo de reconocimiento del color. Requiere condiciones de iluminación estables y la elección de un color de seguimiento muy diferenciado del contexto. **Use Color Tracking**, activa/desactiva su uso y los resultados se ven en el monitor inferior. Como se ha mencionado antes, cuando se trabaja con **COLOR TRACKING** debe subirse al máximo el valor del slider **Substraction Threshold** para anular los datos del **BACKGROUND SUBTRACTION**.

COLOR TRACKING SETTINGS

La configuración de este bloque solo se ajusta si está activado **Use Color Tracking**. El Tracking Color transforma el espacio de color RGB al sistema HSV, porque asigna los valores del tono del color de una forma más estable. HSV identifica cada tono con un número que va de 0ª a 360ª según su posición en el círculo cromático; en GAmuza el rango va de 0 a 1, quedando su equivalencia tal como muestra la imagen.



Las opciones de ajuste son:



Hue, selecciona el tono a trackear según la escala de valores de GAmuza entre 0 y 1. La franja de color sirve de orientación.

Hue Range, amplía el rango del tono seleccionado.

Saturation, asigna el nivel de saturación del tono (hue) seleccionado. El valor 0 es igual a blanco, 1 es el tono puro, dependiendo del nivel de luminosidad (value).

Saturation Range, amplia el rango de saturación seleccionado.

Value, regula el nivel de luminosidad (value) del tono. El valor 0 es igual a negro, 1 es igual al tono puro, dependiendo del valor de saturación.

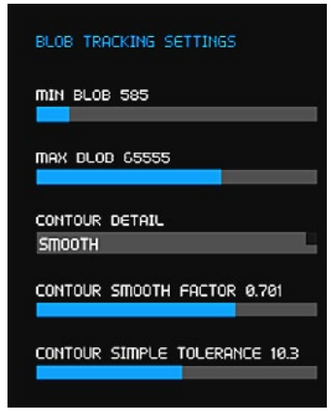
Value Range, amplía el rango de luminosidad seleccionado.

HSV Blur, filtro de desenfoque de la imagen en HSV.

Erode, reduce las superficies detectadas.

Dilate, expande las superficies detectadas.

BLOB TRACKING SETTINGS



Los blob son las zonas independientes que el tracking detecta según los algoritmos seleccionados, sus opciones de configuración son:

Min Blob, tamaño mínimo de los blobs en píxeles.

Max Blob, tamaño máximo.

Contour Detaill, el detalle del contorno se puede ajusta en:

- **Raw**, todo el perímetro tal cual es
- **Smooth**, suavizado
- **Simple**, adaptado a formas geométricas simples

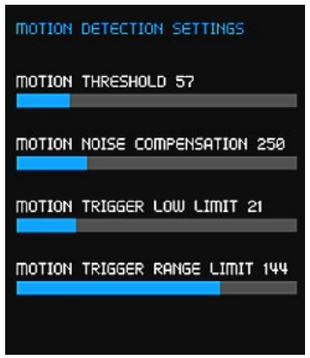
CONTOUR SMOOTH FACTOR, si se ha seleccionado smooth, ajusta el nivel de suavizado

CONTOUR SIMPLE TOLERANCE, si se ha seleccionado simple, ajusta el rango de vértices que articulan el contorno.

MOTION DETECTION SETTINGS

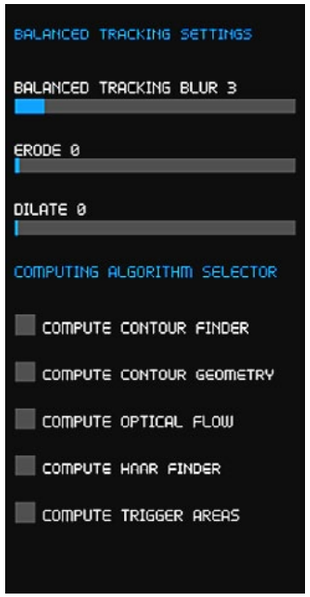
El algoritmo para detección de movimiento está siempre activo.

El monitor **MOTION DETECTION** muestra el cálculo de la cantidad de movimiento que se registra en la imagen y señala la posición media de ese movimiento. Los ajustes para su configuración son:



- **Motion Threshold**, regula la sensibilidad del sistema para activar la detección, si es poco sensible se debe ampliar el umbral (Threshold)
- **Motion Noise Compensation**, compensa el ruido de la señal de imagen video
- **Motion Trigger Low Limit**, nivel más bajo para activar la detección de movimiento
- **Motion Trigger Range Limit**, limita los niveles de movimiento, como si bajara la sensibilidad de detección.

BALANCED TRACKING SETTINGS



Vuelve a ajustar, en conjunto, los valores dados en cada sección.

- **Balanced Tracking Blur**, ajusta el nivel de desenfoque global,
- **Erode**, reduce las superficies detectadas
- **Dilate**, expande las superficies detectadas

COMPUTING ALGORITHM SELECTOR

En este bloque se seleccionan otros algoritmos específicos que se pueden aplicar, además de los mencionados hasta ahora.

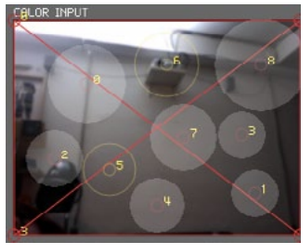
Compute Contour Finder, para procesar el contorno de las zonas detectadas. Sus valores se ajustan con **Blob Tracking Settings**.

Compute Contour geometry, para procesar el contorno de las zonas detectadas con formas geométricas.

Compute Optical Flow, calcula el flujo óptico del movimiento y lo describe con vectores que indican la dirección e intensidad.

Compute Haar Finder, activa el algoritmo para detectar partes del cuerpo humano. Hay distintos algoritmos posibles. Por defecto, está seleccionado **frontalface_alt** (Ver página 225).

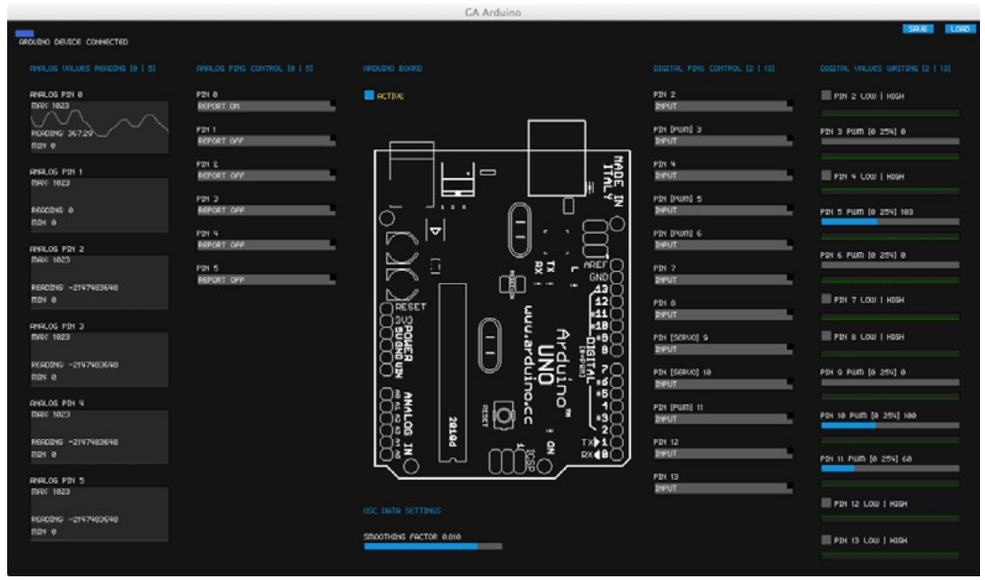
Compute Trigger Areas. Activa la posibilidad de utilizar 9 zonas circulares que tienen asignado un ID. Su posición y tamaño pueden ajustarse con el ratón; con ellas se pueden determinar las áreas en las que debe responder el tracking utilizando funciones específicas de GAmuza.



Si hay varias cámaras conectadas al ordenador, se pueden abrir distintas interfaces del panel para cada una de ellas que pueden ajustarse independientemente, por eso en las funciones de programación hay que señalar el id de la cámara.

6. Interface Módulo Arduino

El módulo Arduino establece la comunicación entre las funciones de GAmuza y el microcontrolador Arduino[18].

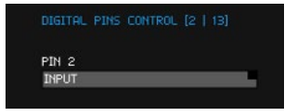


las opciones de configuración son:

ANALOG VALUES RECORDIG. Lectura de valores analógicos, del PIN 0 al 5. Cada monitor muestra el gráfico de la señal y su valor entre 0 y 1023. La función de GAmuza, `ga.analogRead()` normaliza ese rango de valores de 0.0 a 1.0.

En la sección **ANALOG PIN CONTROL**, hay que seleccionar **Report ON** en los pin que se deseen leer y **Report OFF** en los no conectados.

Arduino tiene 14 pin digitales, de los cuales 6 son PWM (Pulse-width modulation) de 8 bits, pudiendo transmitir un rango de valores de 0 a 254. Si se selecciona la opción Input u output funcionan como los demás, solo envían o reciben dos datos: HIGH o LOW.



18 Arduino < http://www.arduino.cc/> [25.08.2012]

Digital values wrting: los ajustes de los valores de escritura de los pin digitales y PWM, están vinculados a los datos programados con la función de GAmuza `ga.digitalWrite(pwmPin,valor)`, el pin al que haga referencia la función debe estar activado en la interface.