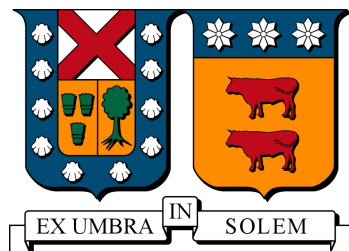


**UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA**  
**DEPARTAMENTO DE ELECTRÓNICA**  
**VALPARAÍSO - CHILE**



**SISTEMA DE ESTIMULACIÓN VISUAL Y  
REGISTRO DE MOVIMIENTOS OCULARES  
PARA TAREAS SICOMOTORAS"**

**CHRISTIAN ANDRÉS WICHE LATORRE**

**MEMORIA DE TITULACIÓN PARA OPTAR AL TÍTULO DE INGENIERO CIVIL  
ELECTRÓNICO MENCIÓN CONTROL E INSTRUMENTACIÓN**

**PROFESOR GUÍA:                   MARÍA JOSÉ ESCOBAR SILVA**  
**PROFESOR CORREFERENTE:    MATÍAS ZAÑARTU SALAS**

**?? - 2018**

# Glosario

**ejes visuales** Corresponde a la proyección de una línea recta que pasa simultáneamente por el centro de la fovea, y la pupila.

**esclerótida** Sección blanca del ojo que rodea a la pupila

**eye tracker** Dispositivo utilizado para realizar seguimiento de los movimientos oculares

**fovea** Área de la retina que permite la visión más nítida y detallada

**retina** Capa fotosensible ubicada en la parte posterior del ojo encargada de transducir estímulos lumínicos en impulsos nerviosos

**setup** Se asigna esta denominación a cierta configuración de un espacio de trabajo

# Siglas

**CRT** Cathode Ray Tube (Tubo de Rayos Catódicos)

**EOG** Electro-OculoGraphy (Electro-oclulografía)

**FP** Fixation Point (Punto de Fijación)

**FPS** Frames Per Second (Cuadros Por Segundo)

**GUI** Graphical User Interface (Interfaz gráfica de Usuario)

**IR** Infra-Red (Infra-Rojo)

**ISI** Inter Saccadic Inteval (Intervalo Inter-Sacádico)

**RT** Response Time (Tiempo de Respuesta)

**SL** Saccadic Latency (Latencia Sacádica)

**SO** Sistema Operativo

**SSC** Scleral Search Coil (Bobina Escleral de Búsqueda)

**TP** Target Point (Punto Objetivo)

**VOG** Video-OculoGraphy (Oculografía basada en Video)

# Índice general

<b>Glosario</b>	<b>I</b>
<b>1. Introducción</b>	<b>1</b>
1.1. Motivación . . . . .	1
1.2. Objetivos . . . . .	2
<b>2. Estado del arte</b>	<b>3</b>
2.1. Movimiento ocular . . . . .	3
2.2. Métodos de captura de movimiento ocular . . . . .	5
2.2.1. Un poco de historia . . . . .	5
2.2.2. Tecnologías actuales . . . . .	6
2.2.3. Comparativa . . . . .	9
2.3. Métodos de estimulación visual . . . . .	10
2.3.1. Hardware de estimulación . . . . .	10
2.3.2. Software de estimulación . . . . .	12
2.3.3. Experimentos de estimulación . . . . .	13
2.4. Sistemas de estimulación y registro visual . . . . .	16
<b>3. Sistema propuesto</b>	<b>18</b>
3.1. <i>Setup</i> a utilizar . . . . .	18
3.2. Diseño del sistema . . . . .	19
3.2.1. Primera parte: Estructura de datos . . . . .	19
3.2.2. Segunda parte: Implementación de las funciones principales . . . . .	22
3.2.3. Tercera parte: Interfaz gráfica . . . . .	37
<b>4. Resultados</b>	<b>38</b>
4.1. Configuración de test de prueba . . . . .	38
4.2. Mediciones obtenidas . . . . .	38
<b>5. Conclusiones y trabajo futuro</b>	<b>39</b>
5.1. Conclusiones . . . . .	39
5.2. Trabajo futuro . . . . .	39
<b>Referencias</b>	<b>40</b>
<b>A. Configuración del entorno de programación</b>	<b>42</b>



# Índice de figuras

2.1. Estructura del ojo . . . . .	3
2.2. Ejemplo de uso de SSG . . . . .	7
2.3. Ejemplo de posicionamiento de electrodos para EOG . . . . .	7
2.4. Principio de detección VOG en base a reflexión de luz . . . . .	8
2.5. Muestra de <i>eye trackers</i> disponibles en el mercado con tecnología VOG basados en reflexión de luz IR . . . . .	9
2.6. Tarea pro/anti sacádica. . . . .	14
2.7. Paradigmas de experimentación. . . . .	15
2.8. Tarea guiada por memoria. . . . .	16
2.9. <i>Setup</i> experimental típico . . . . .	17
2.10. Diagrama general de la interfaz hombre-máquina . . . . .	17
3.1. Diagrama de base de datos a implementar. . . . .	20
3.2. Diagrama de funcionamiento general. . . . .	23
3.3. Diagrama de clases del sistema (no considera GUI). . . . .	24

# Índice de cuadros

2.1. Comparativa de los sistemas de adquisición encontrados . . . . .	10
2.2. Comparativa de software de estimulación . . . . .	13
3.1. Hardware y software utilizado en el desarrollo. . . . .	18
3.2. Métodos implementados en la clase Utils. . . . .	26
3.3. Métodos implementados en la clase SaccadeDB. . . . .	26
3.4. Métodos implementados en la clase Master. . . . .	28
3.5. Métodos implementados en la clase ItemList. . . . .	29
3.6. Métodos implementados en la clase Component. . . . .	31
3.7. Métodos implementados en la clase Frame. . . . .	33
3.8. Métodos implementados en la clase Test. . . . .	34
3.9. Métodos implementados en la clase Experiment. . . . .	36
3.10. Métodos implementados en la clase ExperimentHandler. . . . .	37
3.11. Métodos implementados en la clase ExperimentRuntime. . . . .	37

# Capítulo 1

## Introducción

### 1.1. Motivación

Diariamente y sin prestar mayor atención, gran parte de la población utiliza sus ojos para interactuar con su entorno: se detienen a admirar el paisaje, leer un libro, revisar su teléfono, navegar en internet, verificar que sus alimentos se encuentran en buen estado, etc. A pesar de lo simple que puede parecer esta acción, en la realidad corresponde a un proceso sumamente complejo que involucra la participación de un sinnúmero de estructuras sensoriomotrices.

El simple hecho de orientar nuestra vista hacia un nuevo objetivo desencadena una serie de eventos fascinantes: El globo ocular rota hasta lograr posicionarse en una determinada dirección de forma tal que los rayos de luz que son reflejados por el punto de interés se proyectan en la retina, estructura que transduce esta información en impulsos eléctricos que son interpretados de forma posterior por el cerebro y que se traducen en información que percibimos como una imagen.

El estudio de las dinámicas del ojo y la capacidad de registrar sus movimientos ha permitido con el paso de los años avances importantes en áreas sumamente variadas como la detección y seguimiento de enfermedades, estudios de marketing y usabilidad, ayuda a personas con discapacidad, entrenamiento y detección de errores en sistemas simulados, defensa y seguridad, videojuegos y experiencia de usuario, entre otras.

La principal dificultad en los avances de todas las áreas recae no solo en la calidad del registro ocular tanto en precisión temporal como espacial, si no, en la capacidad de correla-



cionar estímulos visuales y respuestas motrices con el fin de detectar la relación causa/efecto de los eventos. Para lograr esto, es necesario el uso de sistemas que permitan sincronizar las etapas de estimulación y registro de respuesta de forma tal de entregar insumos que permitan la obtención de información pertinente y útil del procesamiento posterior.

## **1.2. Objetivos**

Así, el objetivo principal de este trabajo de título consiste en el diseño y construcción de un sistema de estimulación visual y registro de movimientos oculares enfocado en la realización de experimentos asociados a tareas sicomotoras. De esta forma, se presentan a continuación los objetivos específicos para el desarrollo:

- (I) Definir y programar un mecanismo de estimulación visual acorde con las características técnicas de los protocolos utilizados en la realización de tareas sicomotoras.
- (II) Implementar un sistema de sincronización entre el registro y la estimulación visual que permita la integración de un sistema de captura de movimiento ocular y que asegure el correcto registro de los datos.
- (III) Integrar todas las etapas en una interfaz gráfica de usuario (GUI).

# Capítulo 2

## Estado del arte

### 2.1. Movimiento ocular

La acción de dirigir la mirada hacia un objeto es parte fundamental del proceso de visión. Este acto involucra el direccionamiento de los ejes visuales (ver figura 2.1) hacia un objetivo determinado, permitiendo la realización de análisis visuales precisos. Dicha orientación muchas veces implica movimientos coordinados de los ojos, cuello y cabeza, no obstante, existen movimientos más pequeños que son realizados únicamente por los ojos, conocidos como movimientos sacádicos [1, 2].

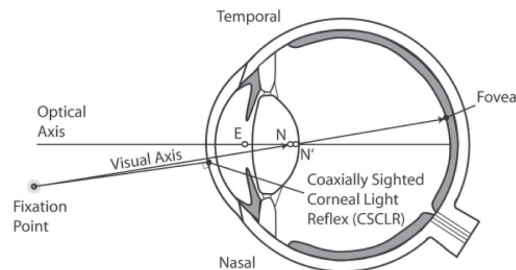


Figura 2.1: Estructura del ojo<sup>1</sup>.

En la figura 2.1 se observa un esquema simplificado del ojo observado desde la parte superior de la cabeza. En ella pueden apreciarse las proyecciones del eje óptico y visual del ojo, que corresponden a líneas imaginarias que van desde el centro de la retina y fóvea

<sup>1</sup>Fuente imagen: J. T. Schwiegerling, "Eye Axes and Their Relevance to Alignment of Corneal Refractive Procedures", Journal of Refractive Surgery, vol 29(8), pp. 515-516, fig. 1-D.

respectivamente, pasando por la parte posterior del cristalino y extendiéndose luego a través de la pupila y córnea.

Los movimientos sacádicos corresponden a las rotaciones que realiza el globo ocular entre dos momentos de posicionamiento estacionario, por tanto se traducen en el consecuente desplazamiento de la pupila. Dichos movimientos pueden ocurrir tanto en el eje horizontal como vertical y se encuentra aún en discusión [2, 3] si se entrega información visual en todo momento o solo es relevante al mantener la mirada detenida en alguna posición. Los movimientos sacádicos ocurren en todo momento y pueden estar provocados por procesos cognitivos conscientes o inconscientes. Independientemente de la naturaleza de estos movimientos, ambos generan patrones que permiten la construcción de un mapa mental de la escena observada. La necesidad de redireccionar el eje visual constantamente para analizar una imagen se relaciona con la idea de que, en el ojo humano, solo la parte central de la retina (fóvea) se encuentra encargada de la visión en alta resolución, pues presenta una alta concentración de células fotorreceptoras sensibles al color.

Existen dos tipos de movimientos sacádicos: Los voluntarios, también conocidos como provocados, que implican control consciente sobre los procesos cognitivos y los reflexivos o automáticos, que corresponden a la respuesta natural a la aparición de un nuevo estímulo visual.

Las características principales de dichos movimientos son:

1. Por su naturaleza los movimientos sacádicos se consideran balísticos, lo que quiere decir que la posición de destino no cambia durante el desarrollo del movimiento. Esto puede entenderse también como que el objetivo se encuentra predeterminado en el momento de partida.
2. La velocidad de las sacadas<sup>2</sup> aumenta de forma no lineal en la medida que aumenta la amplitud de movimiento y puede alcanzar magnitudes de hasta  $600 - 700[\frac{\text{grados}}{\text{s}}]$ . Además, la duración del movimiento puede fluctuar entre  $20 - 120[ms]$ , aunque en promedio solo dura de  $20 - 40[ms]$ .
3. La precisión del movimiento sacádico presenta un error que varía entre el  $5 - 10\%$  de la amplitud total del movimiento. Las correcciones son realizadas por desplazamientos

---

<sup>2</sup>Forma de referirse a un movimiento sacádico.

de calibración denominados micro-sacadas, que permiten además realizar tareas que requieren de gran precisión visual [4]. Estos métodos correctivos permiten suponer que existe algún tipo de procesamiento paralelo encargado de la calibración ocular de largo plazo [2].

Las características expuestas entregan, a grandes rasgos, nociones que permiten comprender el por qué su estudio se ha vuelto común en campos científicos como la neurociencia: Dado que los movimientos del globo ocular son caracterizables, de patrones definidos y de alta precisión es posible identificar mediante ellos enfermedades cuyos síntomas se traduzcan en alteraciones de las capacidades motrices. Un ejemplo de esto es la enfermedad de Parkinson, donde una afección crónica a los ganglios basales produce una reducción progresiva de la sustancia negra lo que se traduce en una producción insuficiente de dopamina, neurotransmisor relevante para la función motora. Esta insuficiencia se traduce en aumentos en los tiempos de respuesta y tasas de error en diversas tareas asociadas a movimiento ocular [5, 6, 7, 8] (ver 2.3.3).

## **2.2. Métodos de captura de movimiento ocular**

### **2.2.1. Un poco de historia**

Para poder registrar los movimientos oculares es necesario el uso de equipamiento especializado que, por su función y sin importar la tecnología utilizada, se denomina por su nombre en inglés: *eye tracker*. A modo introductorio se presenta a continuación una pincelada de su desarrollo en la historia [9, 10]

La primera aparición de dispositivos de este tipo data de finales del siglo XIX [11, 12] y permitieron objetivizar las investigaciones de comportamiento existentes. Sus primeras versiones consistieron en sistemas sumamente invasivos donde alambres finos conectados a una especie de lente de contacto movían una serie de palancas que amplificaban el movimiento y lo registraban en papel. Por su construcción, dichos dispositivos permitían observar el comportamiento espacial, mas no el temporal.

A principios del siglo XX comenzaron a desarrollarse sistemas más parecidos a las tec-

nologías actuales de la mano de técnicas no invasivas basadas en óptica y reflexión de luz: La proyección de luz sobre la córnea genera reflejos que se mueven de forma similar a la pupila. Si se hace posible su registro, es posible conocer el movimiento ocular tanto horizontal como vertical, más no rotatorio [13]. Este método revolucionario marcaría los desarrollos futuros en esta área.

En la década de los 70', gracias a los avances en sistemas de grabación de video y procesamiento digital, se hizo posible detectar electrónicamente características del movimiento en base al contraste existente entre la esclerótica y los bordes del iris. Debido a los efectos de sombra producidos por los párpados, este método presentaba problemas para detectar movimientos verticales, no obstante, permitía registrar movimiento horizontal con buena calidad.

De forma posterior y en base a estos avances iniciales fueron desarrolladas las tecnologías que, cada vez más, permiten obtener información relevante sin producir daño sobre quienes forman parte de los experimentos, reduciendo así las limitantes en este campo de investigación.

### **2.2.2. Tecnologías actuales**

En la actualidad existen una gran gama de tecnologías para registrar movimiento ocular [14, 9, 10]. A continuación se indican las más relevantes:

1. **Bobina escleral magnética (SSC):** Esta técnica requiere del uso de lentes de contacto de gran tamaño directamente sobre el globo ocular. Dicho lente posee dos pequeñas bobinas de alambre que, al ser alineadas con el eje de visión e inducidas por campos electromagnéticos externos de alta frecuencia, permiten obtener información sobre la dirección en la que se encuentra el ojo en forma de voltaje. A pesar de la gran incomodidad que producen (ver figura 2.2), esta técnica es una de las más precisas y exactas.

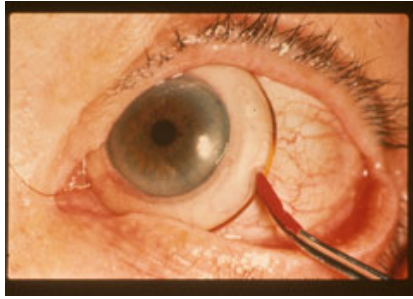


Figura 2.2: Ejemplo de uso de SSG<sup>3</sup>.

2. **Electro-OculoGrafía (EOG):** Este método de seguimiento ocular permite determinar la dirección en la que se encuentra direccionado el ojo en base a la medición de diferencias de potencial eléctrico en la piel. En la medida que el ojo rota, el dipolo producido por la córnea y la retina cambia su orientación, lo que se ve reflejado en los potenciales de las zonas aledañas. Sus ventajas principales, además del bajo costo, son la capacidad de medir el movimiento ocular incluso aunque los ojos se encuentren cerrados (lo que hace de este método una herramienta interesante en casos de estudio de sueño) y que las mediciones son relativas a la posición de la cabeza. No obstante lo anterior, la precisión y exactitud de las mediciones obtenidas es baja ya que se encuentran sujetas a artefactos como el movimiento de los párpados. La figura 2.3 muestra el posicionamiento típico de los electrodos en este tipo de *setup*.

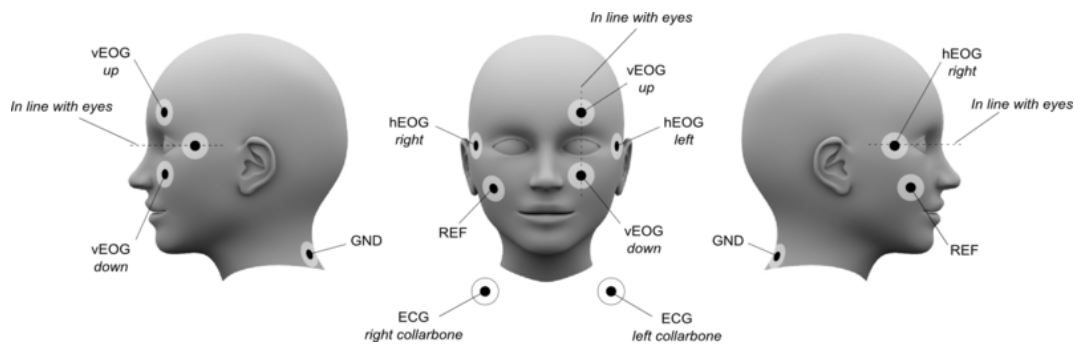


Figura 2.3: Ejemplo de posicionamiento de electrodos para EOG<sup>4</sup>.

3. **Seguimiento ocular basado en video (VOG):** Este método de seguimiento ocular es el más popular en la actualidad. Consiste en capturar, con una o más cámaras, la actividad

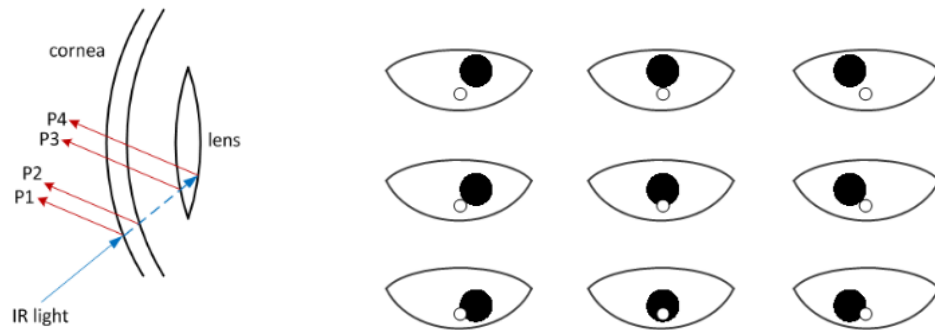
<sup>3</sup>Fuente imagen: <http://www.dizziness-and-balance.com/practice/default.htm>

<sup>4</sup>Fuente imagen: [http://sbiwiki.cns.ki.se/mediawiki/index.php/Natmeg/checklist\\_MEG\\_preparation](http://sbiwiki.cns.ki.se/mediawiki/index.php/Natmeg/checklist_MEG_preparation)

ocular y mediante procesamiento de imágenes determinar en que dirección se encuentra orientada la mirada. El procesamiento puede ser dividido en dos etapas principales: en la primera se detecta y localiza el ojo en la imagen, para lo que típicamente se utiliza la pupila o el iris y la segunda etapa corresponde al proceso por el cual se estima hacia donde se encuentra dirigido.

Dentro de las técnicas VOG existentes la más utilizada corresponde a la estimación mediante reflexión corneal (con una fuente de luz típicamente infrarroja (IR)) y detección de pupila. Los focos de luz se ubican de forma que siempre se encuentren en la misma posición respecto del ojo, con lo cual la reflexión se mantiene virtualmente siempre en el mismo lugar. De esta forma la estimación se genera en base a la diferencia entre la ubicación de centro de la pupila y la posición de la reflexión.

El principio de funcionamiento asociado a este método se basa en las imágenes de Purkinje-Sanson que describen la existencia de al menos 4 reflexiones de luz producidas por la córnea y el cristalino (figura 2.4 (a)) que pueden ser utilizadas como referencia para estimar el direccionamiento del ojo. En la figura 2.4 (b) es posible observar la primera reflexión de Purkinje en relación a distintas posiciones de la pupila.



(a) Reflexiones de Purkinje-Sanson (b) Posición de la reflexión P1 relativa al centro de la pupila.

Figura 2.4: Principio de detección VOG en base a reflexión de luz[14].

La presentación de estos dispositivos es variada y van desde cascos y lentes hasta trípodes y columnas donde se integran un elemento apoya-baribilla y el *eye tracker*, como en la figura 2.9. En la figura 2.5 se muestra un par a modo de referencia.



(a) Tobii Pro Glasses 2



(b) MiraMetrix S2

Figura 2.5: Muestra de *eye trackers* disponibles en el mercado con tecnología VOG basados en reflexión de luz IR<sup>5</sup>.

### 2.2.3. Comparativa

En la literatura consultada [14, 9, 10] se entregan nociones sobre las capacidades operativas que se resumen en la tabla 2.1 para EOG, SSG, VOG (común) y DPI (VOG de gama alta utilizando distintas reflexiones de Purkinje).

En esta, es posible evidenciar a grandes rasgos la cantidad de parámetros a considerar en el proceso de elección de una tecnología particular: Debe existir un compromiso entre la comodidad para el usuario, las prestaciones tanto en precisión temporal como espacial, dificultad para obtener una buena calibración, etc. De forma tal que permitan la ejecución de cierto tipo de experimentos.

Si lo que interesa medir corresponde al efecto de, por ejemplo, las micro-sacadas en el proceso visual, la principal preocupación se centra en obtener una alta precisión espacial y temporal con el fin de caracterizar correctamente estos movimientos. Por otro lado, si lo que interesa estudiar son los patrones de comportamiento de usuarios para un estudio de usabilidad<sup>6</sup> en un sitio web, estas variables no son tan importantes como la comodidad del usuario, ya que sus acciones deben corresponder lo más posible a una situación común, con el fin de reflejar correctamente sus intereses.

<sup>5</sup>Fuente imagen: <https://www.tobii.com/> y <https://www.microway.com.au/> respectivamente.

<sup>6</sup>Estudios enfocados en caracterizar la experiencia de usuario e identificar el comportamiento del mismo en alguna plataforma (pagina web, aplicación, etc.).



	EOG	SSG	VOG	DPI
Precisión espacial (grados)	$\approx 0,5$	$\approx 0,01$	$\approx 0,05$	$\approx 0,017$
Precisión temporal <sup>7</sup> (Hz)	40	500	50 – 400	500 – 1000
Registro de mov. verticales	Es posible pero se encuentran sujetos a error por efecto de artefactos producidos por el párpado	Si	Si	Si
Registro de torsión	No	Si	Si	Si
Tiempo de <i>setup</i>	Lento debido a que requiere la utilización de electrodos	Lento	Rápido	Rápido
Requiere calibración por enfoque	Si	No	Si	Si
Complejidad de calibración	Requiere configuración bitemporal	La no-linealidad puede ser compensada con un modelo basado en ajuste de parámetros	Buena linealidad	Buena linealidad
Invasividad	Electrodos cercanos al ojo (sin contacto), no afecta el campo visual	Lentes de contacto, posibles efectos negativos en precisión visual, incomodidad	Aparato montado en la cabeza (sin contacto con los ojos), limitación moderada del campo visual	Cabeza inmovilizada en un soporte de barbilla, limitación moderada del campo visual

Cuadro 2.1: Comparativa de los sistemas de adquisición encontrados en [14, 9, 10].

## 2.3. Métodos de estimulación visual

### 2.3.1. Hardware de estimulación

Debido a que la presentación de resultados de investigación científica requieren de que los datos obtenidos sean comprobables y los experimentos reproducidos, es que se hace necesario declarar de forma precisa las características de los estímulos, métodos utilizados para adquirir datos y técnicas aplicadas en el procesamiento posterior. Por lo cual, la elección del artefacto de estimulación es una tarea particularmente sensible [15].

---

<sup>7</sup>Frecuencia de muestreo.

A lo largo de la historia, los investigadores dedicados al estudio del movimiento ocular han usado diversas tecnologías con el propósito de estimular visualmente a sus pacientes, no obstante, no fue hasta la década de los 70' y de la mano de los computadores, que los monitores CRT revolucionaron este campo de investigación convirtiéndose en el estándar durante décadas. El motivo principal de su popularidad dice relación con la capacidad que brindan, mediante la programación en computador, de diseñar con relativa facilidad una gran gama de pruebas y experimentos distintos, lo que permitió explorar de forma rápida nuevos métodos e hipótesis. Esto, complementado a la integración del control de los parámetros de estimulación y almacenamiento de resultados e historiales en el mismo dispositivo permitió robustecer los procesos de investigación debido a la capacidad de repetir los experimentos sin afectar las características de los estímulos.

Las limitantes de los primeros dispositivos se fueron subsanando con el avance de la tecnología, de esta forma, los monitores análogos avanzaron hasta alcanzar tasas de refresco elevadas ( $\geq 200[Hz]$ ), una gran gama de colores, buena resolución espacial (alcanzando hasta  $1600[px]$  de ancho), un rápido decaimiento del fósforo de la pantalla que se traduce en tiempos de repuesta reducidos ( $< 1[ms]$ ) y un buen tamaño (típicamente  $20[pulg]$  en la diagonal). En base a esta información pueden definirse dos conceptos relevantes para el hardware de estimulación:

1. **Tasa de refresco (FPS):** Dice relación con la cantidad de veces que se actualiza la imagen de la pantalla por cada segundo. Influye en el timing de los estímulos.
2. **Tiempo de respuesta (RT):** Es cuanto demora un pixel de la pantalla en cambiar su color de blanco a negro. Influye en la calidad y nitidez de las imágenes.

A pesar de la mejoras considerables en sus características, las nuevas tecnologías han hecho desaparecer a los monitores CRT del mercado, siendo más comunes ahora monitores LCD, LED y oLED que tienen pantallas de mayor tamaño, menor consumo de energía, menor radiación electromagnética y una menor huella de carbono. Es importante destacar que, a pesar de que el aspecto de las nuevas tecnologías es similar, sus principios de funcionamiento, capacidades y características difieren.

Existen varias consideraciones que hacer al utilizar estas tecnologías [16, 17]. Transversalmente se tiene un problema de *timing* debido a las bajas tasas de refresco de la mayor parte

de los monitores modernos, lo que vuelve una tarea no trivial el cumplir con los requerimientos de los estímulos y evitar que el usuario perciba una falta de fluidez en la presentación, lo que podría producir que distraiga su atención. En este sentido, por ejemplo, aunque muchos monitores modernos indican que su tasa de refresco se encuentra entre  $60 - 75[Hz]$ , no se aclara en sus hojas de datos cual es el límite efectivo. Este punto se vuelve crítico si se considera que una buena parte de los equipos modernos incorporan sistemas de procesamiento de imágenes para mejorar la calidad, lo que retrasa aún mas estos tiempos.

Otro elemento de cuidado es el tiempo de respuesta. Si este es elevado, producirá un efecto de halos o rastros de luz cuando las imágenes se mueven con rapidez, por esto, es ideal asegurar que este sea reducido para lograr que el cambio entre imágenes no sea notorio y afecte el experimento (esto se hace más presente en casos en los que se muestra secuencias de video).

Finalmente, es importante compaginar las características del experimento con las propiedades lumínicas del equipo ya que es sabido que en tecnologías como la de los monitores LCD tanto el ángulo del observador respecto de la pantalla como las distintas zonas de la misma afectan el color/contraste observado (efecto de retro-iluminación).

### **2.3.2. Software de estimulación**

Tal como se indicó en el apartado anterior la generación de estímulos es una pieza clave en la realización de experimentos ya que permiten preparar un escenario adhoc para la obtención de datos específicos. En el sitio web de Hans Strasburger [18] es posible encontrar una larga lista de software especializado para el desarrollo de experimentos en el área de la psicofísica además de referencias e información sobre sus características. A modo de resumen se presenta en el cuadro 2.2 una comparación entre algunas de las aplicaciones indicadas en la página<sup>8</sup>.

---

<sup>8</sup>Esta selección fue realizada en base al número de citas para cada aplicación en Google Scholar.

	PsychoPy	VissionEgg	PsychoToolbox	Stimulus Presentation
Tipo de software	Open source			Privativo, de pago
Plataforma	python + OpenGL		Matlab/Octave	Software independiente (IDE con editor de python)
Sistema operativo	Linux, MacOS, Windows			Windows
Fecha de última actualización	<i>Dec/2017</i>	<i>Sep/2014</i>	<i>Oct/2017</i>	<i>Abr/2017</i>
Citaciones en Google Scholar	2220	427	6310	3520
Programable	Si			
Imágenes y Video	Si			
Sonido	Si			
Soporte para <i>Eye trackers</i> incluido	Si	No	Si	Si
Capacidad para registro de data	Si	No	Si	Si
Documentación/Foro disponible	Si	No (Link caído)	Si	Si

Cuadro 2.2: Comparativa de software de estimulación [19, 20, 21, 22].

### 2.3.3. Experimentos de estimulación

Debido a la versatilidad de los movimientos sacádicos, se ha desarrollado con el tiempo un número importante de tareas sicomotoras para probar distintos mecanismos cognitivos. Por este motivo y a modo de acotar las actividades asociadas a este trabajo de título, se desarrollará el sistema propuesto de forma tal que permita solo la implementación de experimentos con características similares a los presentados en [5, 6, 7, 8], que corresponden a métodos utilizados en la evaluación y caracterización de desempeño sicomotor en pacientes que padecen la enfermedad de Parkinson. Dichas tareas corresponden a la evaluación de movimientos pro-sacádicos, anti-sacádicos y aquellos guiados por memoria.

En la explicación de estos experimentos se utilizarán dos elementos importantes. El primero corresponde al punto de fijación o FP y el segundo al punto objetivo o TP. FP es un punto ubicado en el centro de la estimulación a modo de referencia y se utiliza para calcular las amplitudes de los movimientos oculares. TP es un punto objetivo que sirve como guía en la realización de los movimientos.

## Métricas de interés

Las métricas principales a considerar para este tipo de experimentos son:

1. **Latencia sacádica (SL):** Lapso de tiempo que transcurre entre la aparición de un nuevo estímulo y el comienzo de una sacada  $[ms]$ .
2. **Intervalo inter-sacádico (ISI):** Tiempo entre el término de un movimiento sacádico y el comienzo de otro en  $[ms]$ .
3. **Ganancia inicial de movimiento:** Cociente entre la amplitud de la sacada y la distancia desde el punto de partida al objetivo.
4. **Tasa de error:** Relación entre el número de veces que el movimiento ejecutado fue errado y el total de movimientos realizados.

## Movimientos pro/anti sacádicos

Aunque se parecen en estructura, el enfoque de estas tareas es completamente distinto. La medición de la respuesta prosacádica tiene como fin el cuantificar la capacidad del individuo para responder de forma reflexiva frente a un nuevo objetivo y entrega información que suele utilizarse como base para comparar otros métodos. Por otro lado, la actividad antisacádica implica la inhibición del comportamiento reflexivo y la realización de un movimiento voluntario en la dirección opuesta, de esta forma la información obtenida permite conocer el estado de estructuras cerebrales como el lóbulo frontal, en donde se procesan las funciones ejecutivas.

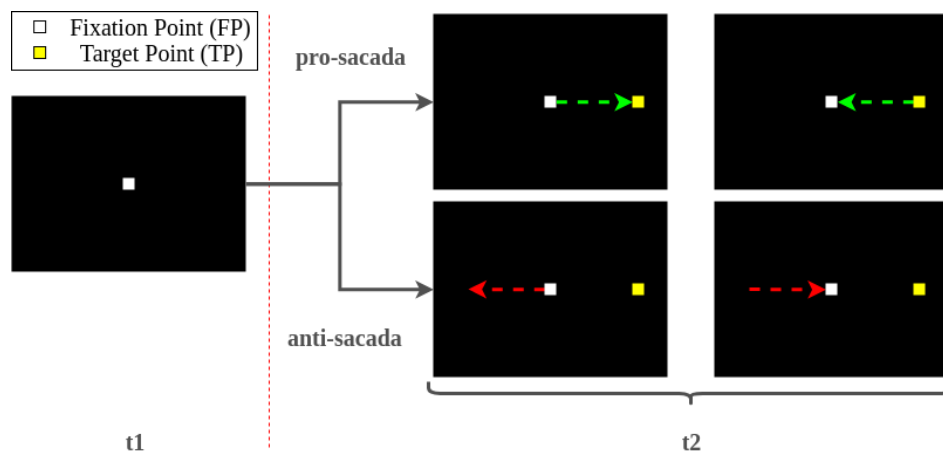


Figura 2.6: Tarea pro/anti sacádica.

Los tipos de movimientos descritos se estudian mediante estímulos similares al propuesto en la figura 2.6. Usualmente se pide al sujeto que mantenga su vista en *FP* (blanco) durante algún tiempo  $t_1$  hasta que aparezca *TP* (amarillo) y luego realice la tarea solicitada, para lo cual se otorga un tiempo  $t_2$ . En el caso de los movimientos prosacádicos, como ya fue explicado con anterioridad, se debe rotar los ojos hacia *TP* y luego volver a *FP* (flechas verdes). Para movimientos antisacádicos se realiza la acción contraria (flechas rojas). En algunos casos resulta conveniente añadir una imagen de feedback al final del experimento para indicar si la tarea fue realizada correctamente.

Estos experimentos pueden ser modificados y complementados con el fin de estudiar actividades cognitivas más complejas. Típicamente estas variaciones aplican uno o más paradigmas como los que se muestran en la figura 2.7, donde puede apreciarse un diagrama de temporización de un estímulo bajo el efecto de los paradigmas de gap, overlap y delay.

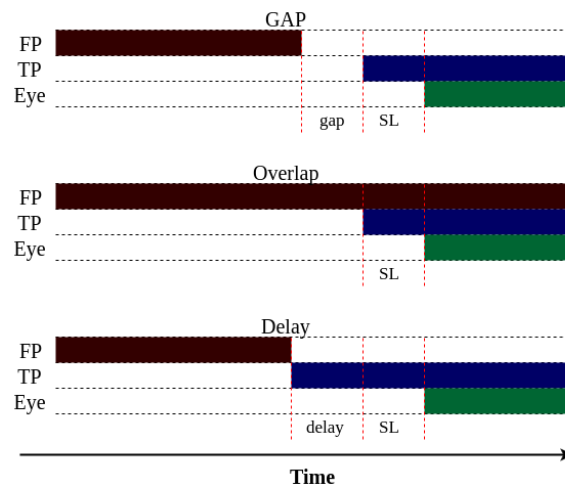


Figura 2.7: Paradigmas de experimentación.

Donde *FP* es el Fixation Point, *TP* el Target Point, *Eye* representa si existe o no movimiento ocular (o más bien, cuando se espera que ocurra) y *SL* corresponde a la latencia sacádica.

### Movimientos guiados por memoria

Los experimentos guiados por memoria corresponden a tareas en las cuales el usuario debe repetir con movimientos oculares algún patrón observado con anterioridad.

En la figura 2.8 es posible observar un diseño simple para este tipo de experimentos.

Después de mantener  $FP$  por un lapso  $t_1$  en pantalla se muestra una serie de objetivos  $T_n$  en intervalos de tiempo idénticos, luego se muestra nuevamente  $TP$  por un lapso  $t_3$  para indicar que ha finalizado la serie y dar tiempo al usuario para prepararse. Finalmente se solicita que se muevan los ojos en secuencia por los puntos en que se vio aparecer los objetivos.

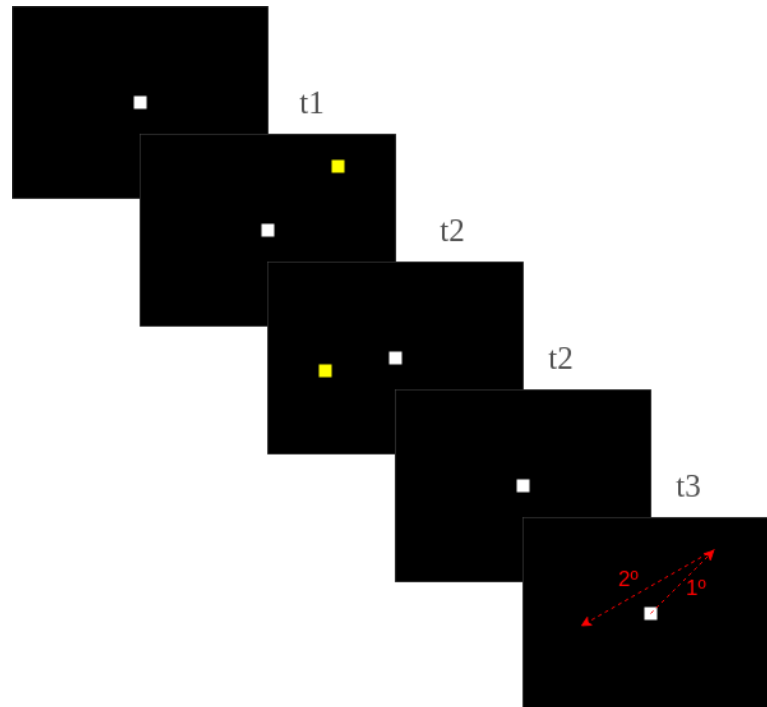


Figura 2.8: Tarea guiada por memoria.

## 2.4. Sistemas de estimulación y registro visual

A continuación se muestran los elementos principales requeridos para la configuración y puesta en marcha de un sistema de estimulación visual y registro de movimiento ocular según lo propuesto por Scott MacKenzie en [23].

En la figura 2.9 se presenta a modo de ejemplo un *setup* típico en estudios de movimiento ocular. Su configuración consiste en un monitor LCD, un *eye tracker* de trípode que en este caso se encuentra montado en un apoya-barbilla y un teclado en el caso de requerir interacción con el usuario. El apoya-barbilla cumple dos funciones sumamente importantes: Mantiene al usuario a una distancia determinada de la pantalla, lo que permite generar estímulos en un rango visual específico y ayuda al paciente a no mover su cabeza durante el experimento, con

lo cual se evitan distorsiones en las mediciones realizadas por el *eye tracker*.



Figura 2.9: *Setup* experimental típico [23].

En la figura 2.10 se pretende explicitar la función del sistema de estimulación - adquisición - registro y los requerimientos del mismo. El software asociado al sistema debe ser capaz de manejar la interfaz con el usuario para poder entregar estímulos (visuales principalmente) y recibir las respuestas asociadas (en forma de datos de posicionamiento ocular o respuestas solicitadas por pantalla) de forma de agrupar las acciones temporalmente, permitiendo de esta forma dar sentido a los datos obtenidos en el experimento.

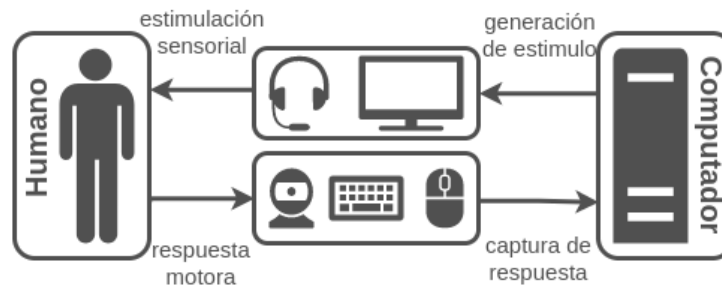


Figura 2.10: Diagrama general de la interfaz hombre-máquina [23].



# Capítulo 3

## Sistema propuesto

### 3.1. Setup a utilizar

Debido a que este trabajo de memoria corresponde fundamentalmente a una prueba de concepto se decidió junto al profesor guía evitar en lo posible la compra de nuevo equipamiento y por tanto no se entrega particular importancia a cumplir de forma estricta con las características de hardware utilizado en instancias formales de investigación. Con esto en consideración se presenta a continuación una descripción del entorno de hardware y software utilizado:

Hardware	PC	Procesador: Intel Core i7-3770 Memoria RAM: 16,0[GB], DDR3, 650[MHz] Disco duro: Samsung SSD, 840 Series, 120[GB] Tarjeta gráfica: NVIDIA GeForce GTX 660 Ti, 2[GB]
	Monitores	Principal: Samsung SyncMaster, 1680 × 1050[px], 60[Hz] Secundario: Samsung SyncMaster, 1680 × 1050[px], 60[Hz]
	Adquisición	Eye tracker: EyeTribe
Software	Entorno	SO: Windows 10 Pro x64 Base: Python 2.7 x86, Anaconda
	Módulos	Principal: PsychoPy 1.84.2[24] Requeridos: Ver anexo A

Cuadro 3.1: Hardware y software utilizado en el desarrollo.

La elección de software se encuentra influenciada fuertemente por los siguientes puntos:

1. El *eye tracker* a utilizar solo tiene disponible sus drivers para Mac y Windows. Debido a la facilidad de encontrar computadoras con Windows se decanta por este SO.
2. Los desafíos del sistema a implementar requieren del uso de un lenguaje orientado a objetos. A pesar de las bondades de Octave o Matlab en procesamiento y acceso a módulos complementarios estos entornos carecen de buen soporte para este tipo de programación, por este motivo se utiliza Python como lenguaje de desarrollo.
3. Dentro de las opciones disponibles PsychoPy resulta sumamente interesante. Cumple con ser open-source, es ampliamente utilizado por investigadores, presenta documentación detallada y clara para todas sus funciones y los foros se encuentran activos. Otro punto interesante es que entrega soporte para varias marcas de *eye tracker*, entre los cuales se encuentra el seleccionado.

## **3.2. Diseño del sistema**

El sistema a ser desarrollado en este trabajo de título tiene como función principal facilitar el proceso de configuración y puesta en marcha de experimentos asociados a movimiento ocular. Para esto, se considera oportuno subdividir el desarrollo en tres partes: La primera consiste en determinar la estructura de datos requerida para almacenar tanto las configuraciones del sistema como de las tareas a implementar. La segunda implica la implementación de los métodos de configuración y ejecución del experimento, asegurando su correcto funcionamiento. La tercera etapa corresponde a montar estas funciones en una GUI para facilitar el proceso a personas que no tengan conocimiento del lenguaje. Así, se presenta a continuación el trabajo realizado.

### **3.2.1. Primera parte: Estructura de datos**

Al diseñar la estructura de datos se debió considerar el problema desde tres aristas. La primera correspondía a la necesidad de almacenar la información del conjunto compuesto por

experimento-tareas-cuadros-componentes donde, complementando lo expuesto en el apartado 2.3.3, las tareas corresponden a las actividades específicas que el sujeto de prueba debe llevar a cabo. Los cuadros corresponden a los elementos que conforman dichas tareas y los componentes serán considerados como las figuras e imágenes contenidas en cada cuadro. La segunda dice relación con la información complementaria que se desea incluir para posterior análisis, tal como comentarios del investigador encargado del experimento para una sesión específica o algunas características de quien realizará el experimento, tales como su edad, sexo, color de ojos o si se utilizan lentes durante la ejecución. Finalmente, la tercera arista implica el conservar las configuraciones del equipo y hardware a utilizar para montar el servicio de ejecución asociado al módulo ioHub, que es el componente de PsychoPy que permite sincronizar todos los dispositivos asociados a la ejecución del experimento: teclado, pantalla, *eye tracker*, etc. y almacenar los datos en un archivo.

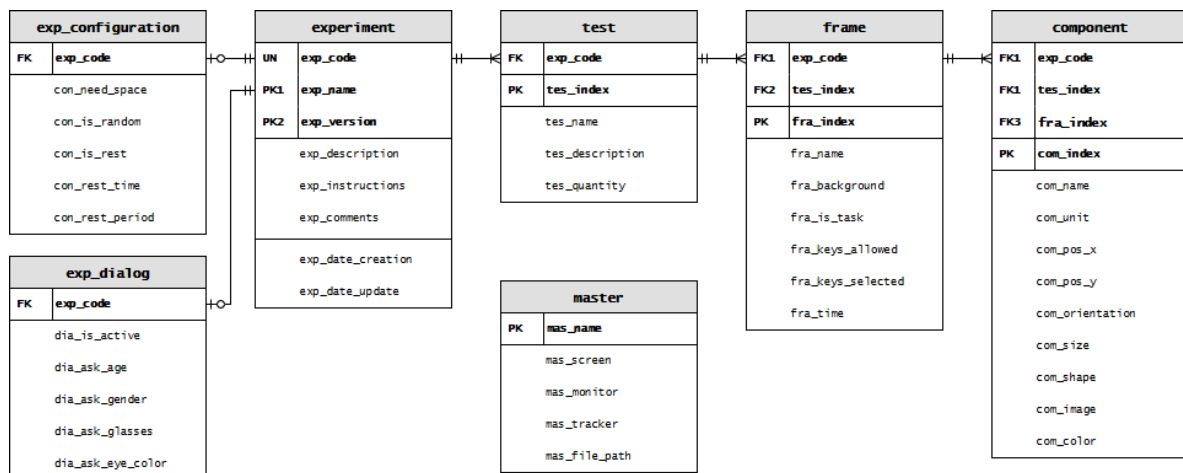


Figura 3.1: Diagrama de base de datos a implementar.

En base a lo anterior, se propone como solución el uso de una base de datos local de tipo SQLite con la estructura presentada en la figura 3.1. A continuación se comenta el contenido de cada tabla y su función:

1. **master:** Almacena la información asociada a un perfil de configuración del hardware a utilizar. Cada perfil es identificado con un nombre e indica qué pantalla será utilizada (screen), cuál es la configuración de dicha pantalla (monitor, se define en un aplicativo de PsychoPy), qué *eye tracker* se utilizará (tracker) y en qué directorio se almacenará

el archivo resultante.

2. **experiment:** Almacena la información de identificación de un experimento. Cada experimento se identifica por un código único (code) y un par nombre/versión (name/-version), que tiene como objetivo identificar a una aplicación específica de una familia del mismo tipo, por ejemplo: "antisaccade"/"v1.0\_gap". Además, se incluye un campo para la descripción, instrucciones para el usuario (instructions) y comentarios para quien tome el experimento (comments). Se incluyen la fecha de creación y la última modificación para dejar de forma explícita una señal de cuidado: Las condiciones del experimento pueden no ser las mismas de la última ejecución.
3. **exp\_configuration:** Esta tabla es complementaria al experimento y almacena la configuración general de ejecución. ¿Es necesario que el paciente presione la tecla espacio antes de cada tarea? (need\_space), ¿El orden de las tareas es aleatorio o secuencial? (is\_random), ¿Se incluyen en el experimento tiempos de descanso? (is\_rest), ¿Cada cuántas tareas? (rest\_period), ¿Con qué duración? (rest\_time).
4. **exp\_dialog:** Esta tabla es complementaria al experimento y almacena la configuración del diálogo inicial, que permite obtener información adicional sobre el paciente. ¿Se incluirá información complementaria? (is\_active), ¿Qué edad tiene el paciente? (ask\_age), ¿Cuál es su sexo? (ask\_gender), ¿Utiliza gafas o lentes de contacto? (ask\_glasses), ¿Cuál es su color de ojos? (ask\_eye\_color).
5. **test:** Esta tabla contiene, para cada experimento, la identificación de las tareas a utilizar. Cada tarea se identifica por su nombre (name) y debe especificar el número de repeticiones a ser ejecutadas (quantity). Incluye un campo para añadir una descripción de la tarea (description).
6. **frame:** Esta tabla contiene, para cada tarea, el conjunto de cuadros que la conforman. Cada cuadro se identifica por un nombre (name) y permite la configuración de su comportamiento y características de forma general: color de fondo (color), si el cuadro corresponde a una tarea que requiere respuesta de teclado o es temporizada

(is\_task), las teclas habilitadas (keys\_allowed), las teclas que se espera sean presionadas (keys\_selected) y el tiempo por el cual debe ser presentada (time).

7. **component:** Esta tabla contiene, para cada cuadro, el conjunto de componentes o elementos que la conforman. Cada componente se identifica por un nombre (name) y las configuraciones asociadas a las unidades de medida consideradas (units), su posición en la pantalla (pos\_x, pos\_y), su tamaño (size), si se encuentra rotada o no (orientation), si es una figura, su forma (shape) y color (color) o si es una imagen el binario correspondiente (image).

Este modelo se considera apropiado por los siguientes motivos:

1. Aísla un experimento de otro al tener tareas y configuraciones completamente independientes. Esta configuración permite evitar que cambios en alguna tarea para ajustarse a necesidades específicas de un experimento afecte el funcionamiento del resto.
2. No puede existir una tarea que no se encuentre asociada a algún experimento, esto aplica también a cuadros y componentes.
3. Tener toda la información y configuraciones contenidas en un archivo facilita la portabilidad y migración desde un equipo a otro. Además, dado que es tratable como una base de datos convencional facilita el proceso de revisión de los datos sin necesidad del programa principal.

Cabe destacar que, para asegurar la limpieza de la base de datos se tomo la decisión de generar tanto updates como deletes en cascada a todas las tablas asociadas a un experimento, de esta forma, al eliminar un experimento particular se eliminan también todas sus tareas y configuraciones, evitando data residual. Esto se repite en niveles más bajos también: eliminar una tarea elimina todos sus cuadros, eliminar un cuadro elimina también todos sus componentes.

### 3.2.2. Segunda parte: Implementación de las funciones principales

Para lograr sincronizar la ejecución del experimento con el uso de dispositivos tales como el monitor, el teclado y el *eye tracker* se hará uso de ioHub [25]. IoHub es un módulo que

forma actualmente parte de PsychoPy y que fue desarrollado originalmente por Sol Simpson. Su funcionalidad principal consiste en montar un servicio por el cual se hace una revisión constante de los dispositivos de entrada/salida seleccionados, almacenando la información recopilada de forma ordenada en un archivo con formato hdf5, que corresponde a un tipo de diccionario donde se ordenan los datos. Para construir la aplicación en base a esta utilidad se debe generar un conjunto de métodos que permita lograr las funcionalidades presentadas en la figura 3.2.

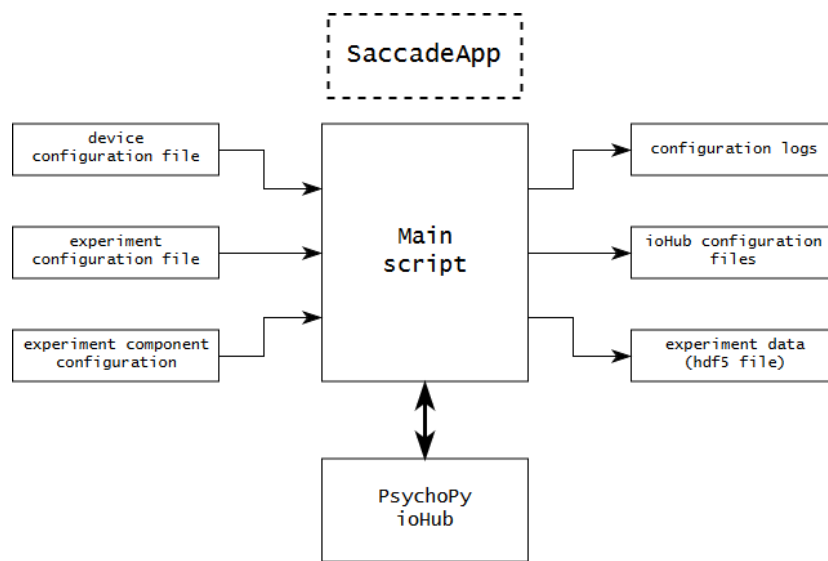


Figura 3.2: Diagrama de funcionamiento general.

Para esto, es necesario entregar tres insumos: la configuración de los dispositivos que serán monitorizados por ioHub, la configuración general del experimento (identificadores, descripción y los elementos que conforman las ventanas de diálogo) y la configuración de los elementos asociados a la presentación de estímulo (que se conforman por otras rutinas de PsychoPy).

En la figura 3.3 se presenta la estructura de clases implementada en este trabajo de título. Las clases Master, Experiment, Test, Frame y Component representan el conjunto de funciones que permiten manejar la configuración almacenada en las respectivas tablas de la base de datos además de otorgar otras funcionalidades que se enfocan en la ejecución del experimento propiamente tal. Debido a que Experiment, Test y Frame implican el manejo de una lista de componentes, estas son creadas como clases que heredan las funcionalidades básicas para

el manejo de listas como el agregar, copiar o eliminar elementos, cambiar su posición en la lista, etc. La clase `utils` implementa métodos de formateo de datos, tales como la conversión de strings a formato unicode o el formateo de fechas. `SaccadeDB` otorga funcionalidades que permiten comunicarse con el archivo de base de datos. Finalmente, la clase `ExperimentHandler` es la encargada de generar crear las carpetas de almacenamiento, los archivos de respaldo, los logs de configuración y la inicialización de la ejecución del experimento, que es implementado en la clase `ExperimentRuntime` (que hereda de `ioHub` e inicia el servicio asociado). La clase `Switch` implementa una estructura similar a un switch-case con el fin de utilizar una máquina de estados que regule la carga de frames durante la ejecución.

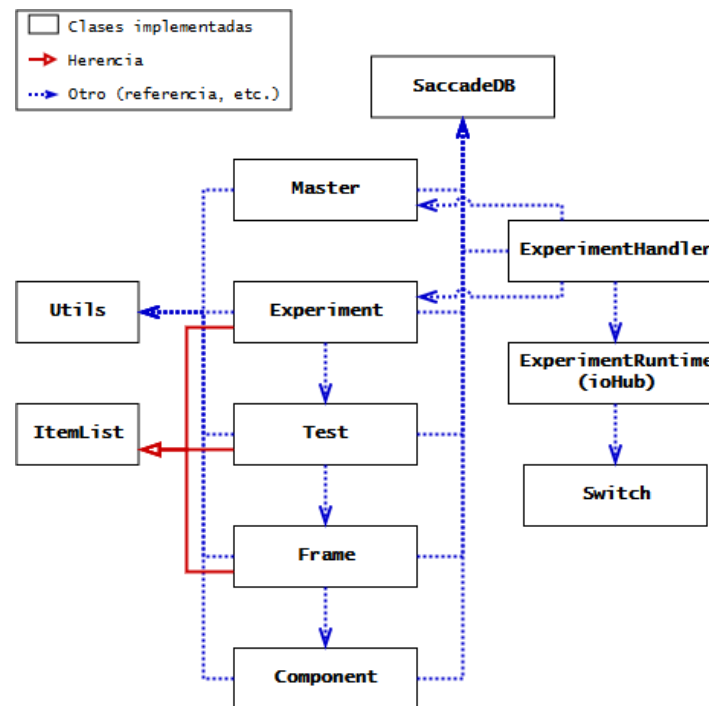


Figura 3.3: Diagrama de clases del sistema (no considera GUI).

A continuación se describe con un mayor grado de detalle cada clase. Por simplicidad se obvian las funciones asociadas al get-set de los atributos (que corresponden a los datos almacenados en la base de datos):

1. **Utils:** Clase utilizada para implementar funcionalidades de formateo de datos.

<b>format_text</b>	
Descripción	Método estático. Convierte un string o similar a formato unicode. Si el dato ingresado no cumple con las condiciones de formato se devuelve un elemento vacío.
Inputs	word: String o dato a ser formateado. lmin: Largo mínimo de la palabra. lmax: Largo máximo de la palabra.
Return	unicode.
<b>format_int</b>	
Descripción	Método estático. Convierte un valor en entero. Si el valor no cumple con las condiciones dadas se devuelve un valor por defecto, especificado por el usuario.
Inputs	value: Valor a ser formateado. vmin: Valor mínimo permitido. default: Valor por defecto.
Return	int.
<b>format_float</b>	
Descripción	Método estático. Convierte un valor en flotante. Si el valor no cumple con las condiciones dadas se devuelve un valor por defecto, especificado por el usuario.
Inputs	value: Valor a ser formateado. vmin: Valor mínimo permitido. default: Valor por defecto.
Return	float.
<b>format_bool</b>	
Descripción	Método estático. Convierte el valor entregado en un booleano. Si el valor no puede ser convertido se devuelve algún valor por defecto especificado por el usuario.
Inputs	state: Estado a ser formateado. default: Valor por defecto.
Return	bool.
<b>format_path</b>	
Descripción	Método estático. Formatea la dirección entregada dependiendo del sistema operativo.
Inputs	path: Dirección a ser formateada.
Return	unicode.



<b>get_time</b>	
Descripción	Método estático. Permite convertir un string asociado a una fecha desde el formato de tiempo GTM a el horario de Chile continental. Esta función se agrega debido a la configuración horaria de la base de datos.
Inputs	date: String que representa una fecha en formato Y-m-d H:M:S.
Return	unicode.

Cuadro 3.2: Métodos implementados en la clase Utils.

2. **SaccadeDB:** Clase utilizada para implementar la conexión con la base de datos SQLite.

<b>SaccadeDB</b>	
Descripción	Constructor. Inicializa los atributos del objeto.
Inputs	filepath: Ubicación (string o unicode) del archivo de base de datos.
Return	void.
<b>connect</b>	
Descripción	Método que inicia una conexión con el archivo de base de datos. Si el archivo de base de datos especificado en el constructor no existe, crea uno nuevo utilizando las configuraciones especificadas en ??.
Inputs	void.
Return	void.
<b>close</b>	
Descripción	Método que finaliza, de ser posible, la conexión con la base de datos.
Inputs	void.
Return	bool. Verdadero si la conexión se cierra apropiadamente, falso en caso contrario.
<b>push_query</b>	
Descripción	Método que permite realizar queries que involucren modificar los contenidos de la base de datos (insert, update, delete, etc.).
Inputs	query. Instrucción SQL (string o unicode).
Return	bool. Verdadero si la query se ejecuta correctamente, falso en caso contrario.
<b>pull_query</b>	
Descripción	Método que permite realizar queries que involucren obtener datos de la base de datos (select).
Inputs	query. Instrucción SQL (string o unicode).
Return	objeto. numpy_array en caso de existir datos, None en caso contrario.

Cuadro 3.3: Métodos implementados en la clase SaccadeDB.

3. **Master:** Clase que permite manipular las configuraciones de hardware asociado a los experimentos y la ruta de los archivos de salida de los mismos. Para utilizar las funciones de carga/guardado es necesario configurar el objeto de base de datos mediante la función set\_database.

Master	
Descripción	Constructor. Inicializa los atributos del objeto.
Inputs	void.
Return	void.
get_list	
Descripción	Método estático. Entrega una lista con los nombres de los perfiles de configuración que se encuentran almacenados en la base de datos.
Inputs	db: Objeto de base de datos saccadeDB.
Return	objeto. list en caso de existir registros, None en caso contrario.
get_available_trackers	
Descripción	Método estático. Entrega una lista de las configuraciones disponibles para los <i>eye trackers</i> . Dichas configuraciones pueden ser encontradas en la carpeta 'saccadeApp/resources/eyetrackers' del proyecto.
Inputs	void.
Return	list.
get_available_screens	
Descripción	Método estático. Entrega una lista de los monitores (hardware) disponibles para realizar el experimento con sus respectivas resoluciones e identificadores. La pantalla principal tendrá el indicador 0.
Inputs	void.
Return	list.
get_available_monitors	
Descripción	Método estático. Entrega una lista de las configuraciones de monitor disponibles. Estas deben ser añadidas en la aplicación Monitor Center de PsychoPy y permiten setear características como la resolución de pantalla, tamaño físico de la misma, distancia del usuario, etc.
Inputs	void.
Return	list.
load	
Descripción	Método que permite cargar en el objeto las configuraciones almacenadas en la base de datos.
Inputs	name: Nombre del perfil de configuración que se desea cargar.
Return	bool. Verdadero si el proceso se completó satisfactoriamente, falso en caso contrario.
save	
Descripción	Método que permite guardar la configuración del objeto en la base de datos.
Inputs	void.
Return	bool. Verdadero si el proceso se completó satisfactoriamente, falso en caso contrario.

copy		
Descripción	Método que permite realizar una copia de la configuración en otro objeto (copia profunda). Esto permite tener distintos perfiles de forma rápida variando pequeños elementos de cada uno.	
Inputs	name:	Nombre del nuevo perfil de configuración. Este no debe haber sido usado antes.
Return	objeto.	Devuelve una copia del objeto actual si se ingresa un nombre correctamente o None en caso contrario.
remove		
Descripción	Método que permite eliminar una configuración de la base de datos.	
Inputs	void.	
Return	bool.	Verdadero si la acción se realizó correctamente, falso en caso contrario.
get_iohub		
Descripción	En base a los atributos de la clase, correspondientes a la tabla master de la base de datos, esta función genera y retorna un diccionario con las configuraciones de los dispositivos requeridos por ioHub.	
Inputs	void.	
Return	dict.	
get_configuration		
Descripción	Retorna un diccionario con los atributos contenidos en la clase. Estos datos se almacenarán en un archivo a modo de log para respaldar e indicar cual fue la configuración utilizada al ejecutar un experimento específico.	
Inputs	void.	
Return	dict.	

Cuadro 3.4: Métodos implementados en la clase Master.

4. **ItemList:** Clase utilizada para manejar listas de objetos. Su implementación no considera el uso directo por parte de los usuarios del programa.

ItemList		
Descripción	Constructor. Inicializa los atributos de la clase y permite definir el tipo de datos que contendrá la lista.	
Inputs	itemclass.	Tipo de dato/clase de los objetos.
Return	void.	
item_add		
Descripción	Método protegido. Permite agregar un objeto a la lista siempre y cuando sea del tipo especificado en el constructor.	
Inputs	item:	Objeto a añadir.
Return	bool.	Verdadero si la acción se realizó correctamente, falso en caso contrario.

<b>item_copy</b>		
Descripción	Método protegido. Agrega una copia profunda del objeto seleccionado en el último lugar de la lista.	
Inputs	index:	Posición en la lista del objeto a copiar.
Return	bool.	Verdadero si el índice existe y se realiza la copia, falso en caso contrario.
<b>item_delete</b>		
Descripción	Método protegido. Elimina el objeto seleccionado.	
Inputs	index:	Posición en la lista del objeto a eliminar.
Return	bool.	Verdadero si el índice existe y es eliminado, falso en caso contrario.
<b>item_move_up</b>		
Descripción	Método protegido. Si es posible, mueve el objeto en un espacio hacia la parte superior de la lista (índices más pequeños).	
Inputs	index:	Posición en la lista del objeto a mover.
Return	bool.	Verdadero si la acción se realizó correctamente (fue posible mover el objeto), falso en caso contrario (no hay lugar o el índice no existe).
<b>item_move_down</b>		
Descripción	Método protegido. Si es posible, mueve el objeto en un espacio hacia la parte inferior de la lista (índices más altos).	
Inputs	index:	Posición en la lista del objeto a mover.
Return	bool.	Verdadero si la acción se realizó correctamente (fue posible mover el objeto), falso en caso contrario (no hay lugar o el índice no existe).
<b>item_get_all</b>		
Descripción	Método protegido. Retorna la lista de objetos.	
Inputs	void.	
Return	objeto.	list en caso de existir objetos, None en caso contrario.
<b>item_get_by_index</b>		
Descripción	Método protegido. Si es posible, devuelve el objeto asociado al índice.	
Inputs	index:	Posición en la lista del objeto seleccionado.
Return	objeto.	Objeto de tipo itemclass si el índice existe o None en caso contrario.
<b>item_number</b>		
Descripción	Método protegido. Devuelve el número de elementos de la lista.	
Inputs	void.	
Return	objeto.	int si existen objetos en la lista o None en caso contrario.

Cuadro 3.5: Métodos implementados en la clase ItemList.

5. **Component:** Clase utilizada para manejar la configuración de los elementos que conforman un cuadro.

Component		
Descripción	Constructor. Inicializa los atributos del objeto.	
Inputs	void.	
Return	void.	
get_list		
Descripción	Método estático. Retorna una lista con todos los componentes que forman parte de un cuadro específico.	
Inputs	db:	Objeto de base de datos (SaccadeDB) mediante el cual se extraerá la información.
	exp:	Código del experimento al que pertenecen los componentes de interés.
	tes:	ID de la tarea a la cual pertenecen los componentes de interés.
	fra:	ID del cuadro al cual pertenecen los componentes de interés.
Return	objeto.	list si existen elementos en la base de datos, None en caso contrario.
encode_image		
Descripción	Método privado. En caso de que el componente corresponda a una imagen, esta función la codifica para almacenar los datos en la base de datos en un campo de tipo BLOB.	
Inputs	void.	
Return	objeto.	Unicode que contiene la imagen codificada en 'base64' en caso de existir imagen, None en caso contrario.
decode_image		
Descripción	Método privado. Permite decodificar una imagen almacenada en 'base64'.	
Inputs	encimg:	unicode que contiene la imagen codificada en 'base64'.
Return	objeto.	Imagen PIL en caso de lograr decodificar, None en caso contrario.
load		
Descripción	Método que permite cargar en el objeto las configuraciones almacenadas en la base de datos.	
Inputs	db:	Objeto de base de datos (SaccadeDB) mediante el cual se extraerá la información.
	exp:	Código del experimento al que pertenece el componente.
	tes:	ID de la tarea a la cual pertenece el componente.
	fra:	ID del cuadro al cual pertenece el componente.
	com:	ID del componente.
Return	bool.	Verdadero si el proceso se completó satisfactoriamente, falso en caso contrario.
save		
Descripción	Método que permite guardar los datos de un componente en la base de datos.	
Inputs	db:	Objeto de base de datos (SaccadeDB) mediante el cual se guardará la información.
	exp:	Código del experimento al que pertenecerá el componente.
	tes:	ID de la tarea a la cual pertenecerá el componente.
	fra:	ID del cuadro al cual pertenecerá el componente.

	com:	ID que se le otorgará al componente.
Return	bool.	Verdadero si el proceso se completó satisfactoriamente, falso en caso contrario.
<b>copy</b>		
Descripción	Método que entrega una copia profunda del objeto.	
Inputs	void.	
Return	objeto.	Objeto de tipo Component con los mismos atributos.
<b>get_execution</b>		
Descripción	Método a ser utilizado durante la ejecución de un experimento. Retorna un objeto que puede ser presentado en el monitor mediante una ventana de PsychoPy.	
Inputs	win:	Objeto de tipo Window perteneciente a PsychoPy que permite mostrar los estímulos por pantalla.
Return	objeto.	Estímulo de la familia 'psychopy.visual'. Actualmente se encuentran implementadas 5 figuras (cruz, cuadrado, círculo, gaussiana y flechas) además de la posibilidad de incluir imágenes.
<b>get_configuration</b>		
Descripción	Retorna un diccionario con los atributos contenidos en la clase.	
Inputs	void:	
Return	dict.	

Cuadro 3.6: Métodos implementados en la clase Component.

6. **Frame:** Clase utilizada para configurar el comportamiento de un cuadro específico de una tarea. Ya que los cuadros contienen y muestran componentes, esta clase se define como hija de ItemList inicializada para objetos de tipo Component.

Frame		
Descripción	Constructor. Inicializa los atributos del objeto.	
Inputs	void.	
Return	void.	
get_list		
Descripción	Método estático. Retorna una lista con todos los cuadros que forman parte de una tarea específica.	
Inputs	db:	Objeto de base de datos (SaccadeDB) mediante el cual se extraerá la información.
	exp:	Código del experimento al que pertenecen los cuadros de interés.
	tes:	ID de la tarea a la cual pertenecen los cuadros de interés.
Return	objeto.	list si existen elementos en la base de datos, None en caso contrario.
load		
Descripción	Método que permite cargar el cuadro y sus componentes desde la base de datos. Para cargar los componentes se hace uso del método privado load_components.	
Inputs	db:	Objeto de base de datos (SaccadeDB) mediante el cual se extraerá la información.
	exp:	Código del experimento al que pertenece el cuadro.

	tes: ID de la tarea a la cual pertenece el cuadro. fra: ID del cuadro.
Return	bool. Verdadero si el proceso se completó satisfactoriamente, falso en caso contrario.
<b>load_components</b>	
Descripción	Método privado. Busca en la base de datos todos los componentes asociados al cuadro de interés, lo carga de forma iterativa y los agrega a la lista.
Inputs	db: Objeto de base de datos (SaccadeDB) mediante el cual se extraerá la información. exp: Código del experimento al que pertenece el cuadro. tes: ID de la tarea a la cual pertenece el cuadro. fra: ID del cuadro.
Return	void.
<b>save</b>	
Descripción	Método que permite guardar los datos de un cuadro y sus componentes en la base de datos. Para guardar los componentes se hace uso del método privado save_components.
Inputs	db: Objeto de base de datos (SaccadeDB) mediante el cual se guardará la información. exp: Código del experimento al que pertenecerá el cuadro. tes: ID de la tarea a la cual pertenecerá el cuadro. fra: ID que se le otorgará al cuadro.
Return	bool. Verdadero si el proceso se completó satisfactoriamente, falso en caso contrario.
<b>save_components</b>	
Descripción	Método privado. Guarda en la base de datos cada uno de los componentes de la lista.
Inputs	db: Objeto de base de datos (SaccadeDB) mediante el cual se guardará la información. exp: Código del experimento al que pertenece el cuadro. tes: ID de la tarea a la cual pertenece el cuadro. fra: ID del cuadro.
Return	void.
<b>copy</b>	
Descripción	Método que entrega una copia profunda del objeto.
Inputs	void.
Return	objeto. Objeto de tipo Frame con los mismos atributos.
<b>get_execution</b>	
Descripción	Método a ser utilizado durante la ejecución de un experimento. Retorna un diccionario que contiene las configuraciones del cuadro y una lista con sus componentes. Para esto se hace uso de forma iterativa de la función get_execution de cada componente que forma parte del cuadro.
Inputs	win: Objeto de tipo Window perteneciente a PsychoPy que permite mostrar los estímulos por pantalla.
Return	dict.

<b>get_configuration</b>	
Descripción	Retorna un diccionario que contiene los atributos del cuadro y una lista con los atributos de sus componentes. Para esto se hace uso de forma iterativa de la función get_configuration de cada componente que forma parte del cuadro.
Inputs	void.
Return	dict.

Cuadro 3.7: Métodos implementados en la clase Frame.

7. **Test:** Clase utilizada para configurar una tarea específica en un experimento. Ya que las tareas se componen de cuadros, esta clase se define como hija de ItemList inicializada para objetos de tipo Frame.

<b>Test</b>	
Descripción	Constructor. Inicializa los atributos del objeto.
Inputs	void.
Return	void.
<b>get_list</b>	
Descripción	Método estático. Retorna una lista con todas las tareas que forman parte de un experimento específico.
Inputs	db: Objeto de base de datos (SaccadeDB) mediante el cual se extraerá la información. exp: Código del experimento al cual pertenecen las tareas de interés.
Return	objeto. list si existen elementos en la base de datos, None en caso contrario.
<b>load</b>	
Descripción	Método que permite cargar la tarea y sus cuadros desde la base de datos. Para cargar los cuadros se hace uso del método privado load_frames.
Inputs	db: Objeto de base de datos (SaccadeDB) mediante el cual se extraerá la información. exp: Código del experimento al que pertenece la tarea de interés. tes: ID de la tarea.
Return	bool. Verdadero si el proceso se completó satisfactoriamente, falso en caso contrario.
<b>load_frames</b>	
Descripción	Método privado. Busca en la base de datos todos los cuadros asociados a la tarea de interés, los carga de forma iterativa y los agrega a la lista.
Inputs	db: Objeto de base de datos (SaccadeDB) mediante el cual se extraerá la información. exp: Código del experimento al que pertenece la tarea de interés. tes: ID de la tarea.
Return	void.
<b>save</b>	
Descripción	Método que permite guardar los datos de una tarea y sus cuadros en la base de datos. Para guardar los cuadros se hace uso del método privado save_frames.



Inputs	db: Objeto de base de datos (SaccadeDB) mediante el cual se guardará la información. exp: Código del experimento al que pertenecerá la tarea. tes: ID que se le otorgará a la tarea.
Return	bool. Verdadero si el proceso se completó satisfactoriamente, falso en caso contrario.
<b>save_frames</b>	
Descripción	Método privado. Guarda en la base de datos cada uno de los cuadros de la lista.
Inputs	db: Objeto de base de datos (SaccadeDB) mediante el cual se guardará la información. exp: Código del experimento al que pertenece la tarea. tes: ID de la tarea a la cual pertenecerá el cuadro.
Return	void.
<b>copy</b>	
Descripción	Método que entrega una copia profunda del objeto.
Inputs	void.
Return	objeto. Objeto de tipo Test con los mismos atributos.
<b>get_execution</b>	
Descripción	Método a ser utilizado durante la ejecución de un experimento. Retorna un diccionario que contiene las configuraciones de la tarea y una lista con sus cuadros. Para esto se hace uso de forma iterativa de la función get_execution de cada cuadro que forma parte de la tarea.
Inputs	win: Objeto de tipo Window perteneciente a PsychoPy que permite mostrar los estímulos por pantalla.
Return	dict.
<b>get_configuration</b>	
Descripción	Retorna un diccionario que contiene los atributos de la tarea y una lista con los atributos de sus cuadros. Para esto se hace uso de forma iterativa de la función get_configuration de cada cuadro que forma parte de la tarea.
Inputs	void.
Return	dict.

Cuadro 3.8: Métodos implementados en la clase Test.

8. **Experiment:** Clase utilizada para configurar un experimento específico. Ya que los experimentos se componen de tareas, esta clase se define como hija de ItemList inicializada para objetos de tipo Test. Para utilizar las funciones de carga/guardado es necesario configurar el objeto de base de datos mediante la función set\_database.

<b>Experiment</b>	
Descripción	Constructor. Inicializa los atributos del objeto.
Inputs	void.
Return	void.

get_list		
Descripción	Método estático. Retorna una lista con todas las tareas que forman parte de un experimento específico.	
Inputs	db:	Objeto de base de datos (SaccadeDB) mediante el cual se extraerá la información.
Return	objeto.	list si existen elementos en la base de datos, None en caso contrario.
load		
Descripción	Método que permite cargar el experimento y sus tareas desde la base de datos. Para cargar las tareas se hace uso del método privado load_tests.	
Inputs	code:	Código que identifica al experimento de interés.
Return	bool.	Verdadero si el proceso se completó satisfactoriamente, falso en caso contrario.
load_tests		
Descripción	Método privado. Busca en la base de datos todos las tareas asociados al experimento de interés, las carga de forma iterativa y los agrega a la lista.	
Inputs	void.	
Return	void.	
save		
Descripción	Método que permite guardar los datos de un experimento y sus tareas en la base de datos. Para guardar las tareas se hace uso del método privado save_tests.	
Inputs	void.	
Return	bool.	Verdadero si el proceso se completó satisfactoriamente, falso en caso contrario.
save_tests		
Descripción	Método privado. Guarda en la base de datos cada una de las tareas de la lista.	
Inputs	void.	
Return	void.	
copy		
Descripción	Método que permite realizar una copia de la configuración en otro objeto (copia profunda).	
Inputs	code:	Código del nuevo experimento. Debido a que este es un campo único, no puede ser igual a alguno ya almacenado en la base de datos.
	version:	Versión del nuevo experimento. no pueden existir dos experimentos con el mismo nombre y versión en la base de datos.
Return	objeto.	Objeto de tipo Experiment si ingresa correctamente el nuevo código y versión, None en caso contrario.

<b>get_iohub</b>	
Descripción	En base a los atributos de la clase, correspondientes a las tablas de la base de datos experiment y exp_dialog, esta función genera y retorna un diccionario con las configuraciones del experimento requeridas por ioHub.
Inputs	void.
Return	dict.
<b>get_execution</b>	
Descripción	Método a ser utilizado durante la ejecución de un experimento. Retorna un diccionario que contiene las configuraciones asociadas a la ejecución del experimento y una lista con sus tareas. Para esto se hace uso de forma iterativa de la función get_execution de cada tarea que forma parte del experimento.
Inputs	win: Objeto de tipo Window perteneciente a PsychoPy que permite mostrar los estímulos por pantalla.
Return	dict.
<b>get_configuration</b>	
Descripción	Retorna un diccionario que contiene los atributos del experimento y una lista con los atributos de sus tareas. Para esto se hace uso de forma iterativa de la función get_configuration de cada tarea que forma parte del experimento.
Inputs	void:
Return	dict.

Cuadro 3.9: Métodos implementados en la clase Experiment.

9. **ExperimentHandler:** Clase utilizada para efectuar las configuraciones previas que permiten ejecutar un experimento.

<b>ExperimentHandler</b>	
Descripción	Constructor. Inicializa los atributos del objeto.
Inputs	void.
Return	void.
<b>load_experiment</b>	
Descripción	Método que verifica la existencia del perfil de configuración (Master) y el experimento (Experiment) a ser utilizados. Si todo está correcto habilita la ejecución del experimento.
Inputs	db: Objeto de base de datos (SaccadeDB) mediante el cual se extraerá la información. mas: Nombre dle perfil de configuración a utilizar. exp: Código del experimento a utilizar.
Return	bool. Verdadero si esta habilitada la ejecución, falso en caso contrario.
<b>save_execution_parameters</b>	
Descripción	Si el experimento se encuentra habilitado, este método verifica la existencia de las carpetas en donde se almacenarán los resultados y guarda en ellas respaldos de los archivos de configuración acompañados del timestamp en que fueron generados.
Inputs	void:
Return	bool. Verdadero si el experimento esta habilitado y se guardaron los archivos, falso en caso contrario.

<b>execute_experiment</b>		
Descripción	Si el experimento se encuentra habilitado y los archivos de configuración están disponibles se ejecuta el experimento. Para esto, se inicializa una instancia de la clase <code>ExperimentRuntime</code> .	
Inputs	void.	
Return	bool.	Verdadero si el experimento pudo ser ejecutado, falso en caso contrario.

Cuadro 3.10: Métodos implementados en la clase `ExperimentHandler`.

10. **ExperimentRuntime:** Clase que ejecuta el experimento. Hereda de `ioHubExperimentRuntime`, clase de `ioHub` encargada de iniciar el servicio por el cual se manejan todos los dispositivos de hardware.

ExperimentRuntime		
Descripción	Constructor. Inicializa los atributos de la clase y permite entregar la configuración del experimento a la rutina.	
Inputs	exp_cfg_dict.	Diccionario que almacena el experimento a ejecutar y la ubicación de los archivos de configuración necesarios para correr el servicio.
Return	void.	
run		
Descripción	Método reimplementado de la clase padre y que tiene por función definir tanto la inicialización de los dispositivos como el script del experimento a ejecutar. Es importante destacar que este método NO se ejecuta directamente, para ello se utiliza la función específica ExperimentRuntime.start.	
Inputs	*args:	No utilizado.
Return	void.	

Cuadro 3.11: Métodos implementados en la clase `ExperimentRuntime`.

### 3.2.3. Tercera parte: Interfaz gráfica

Pendiente.

## **Capítulo 4**

### **Resultados**

#### **4.1. Configuración de test de prueba**

#### **4.2. Mediciones obtenidas**

## **Capítulo 5**

### **Conclusiones y trabajo futuro**

#### **5.1. Conclusiones**

#### **5.2. Trabajo futuro**

# Referencias

- [1] K. R. Gegenfurtner, “The interaction between vision and eye movements,” *Perception*, vol. 45(12), pp. 1333–1357, 2016. 3
- [2] J. Findlay and R. Walker, “Human saccadic eye movements,” *Scholarpedia*, vol. 7(7):5095, 2012. 3, 4, 5
- [3] M. Rucci and M. Poletti, “Control and functions of fixational eye movements,” *Annual Review of Vision Science*, vol. 1, pp. 499–518, 2015. 4
- [4] H.-k. Ko, M. Poletti, and M. Rucci, “Microsaccades precisely relocate gaze in a high visual acuity task,” *Nature Neuroscience*, vol. 13, pp. 1549–1553, 2010. 5
- [5] C. J. Luek [et al], “Antisaccades and remembered saccades in parkinson’s disease,” *Journal of Neurology, Neurosurgery, and Psychiatry*, vol. 53(4), pp. 284–288, 1990. 5, 13
- [6] F. Chan [et al], “Deficits in saccadic eye-movement control in parkinson’s disease,” *Neuropsychologia*, vol. 43(5), pp. 784–796, 2005. 5, 13
- [7] S. Amador [et al], “Dissociating cognitive deficits involved in voluntary eye movement dysfunctions in parkinson’s disease patients,” *Neuropsychologia*, vol. 44(8), pp. 1475–1482, 2006. 5, 13
- [8] A. Srivastava [et al], “Saccadic eye movements in parkinson’s disease,” *Indian Journal of Ophthalmology*, vol. 62(5), pp. 538–544, 2014. 5, 13
- [9] T. Eggert, “Eye movement recordings: methods,” *Neuro-Ophthalmology*, vol. 40, pp. 15–34, 2007. 5, 6, 9, 10
- [10] D. C. Richardson and M. J. Spivet, “Eye tracking: Characteristics and methods,” *Encyclopedia of biomaterials and biomedical engineering*, vol. 3, p. 10281042, 2004. 5, 6, 9, 10
- [11] E. B. Delabarre, “A method of recording eye movements,” *American Journal of Psychology*, vol. 9(4), pp. 572–574, 1898. 5
- [12] E. B. Huey, “Preliminary experiments in the physiology and psychology of reading,” *American Journal of Psychology*, vol. 9(4), pp. 575–586, 1898. 5

- [13] R. Dodge and T. S. Cline, “The angle velocity of eye movements,” *Psychological Review*, vol. 8, pp. 145–157, 1901. 6
- [14] G. Rakoczi, “Analysis of eye movements in the context of e-learning, recommendations for eye-efficient user interfaces,” Ph.D. dissertation, Fakultät für Informatik der Technischen Universität Wien, 2014. 6, 8, 9, 10
- [15] B. Bauer, “A timely reminder about stimulus display times and other presentation parameters on crts and newer technologies,” *Canadian Journal of Experimental Psychology*, vol. 69, pp. 264–273, 2015. 10
- [16] P. Wang, “An lcd monitor with sufficiently precise timing for research in vision,” *Encyclopedia of biomaterials and biomedical engineering*, vol. 5, 2011. 11
- [17] T. Elze, T. G. Tanner, and B. Krekelberg, “Temporal properties of liquid crystal displays: Implications for vision science experiments,” *PLoS ONE*, vol. 7(9), 2012. 11
- [18] H. Strasburger, “Software for visual psychophysics: an overview,” [consulta: 10 junio 2017]. [Online]. Available: <http://www.visionscience.com/documents/strasburger/strasburger.html> 12
- [19] N. B. Systems, “Presentation software,” [consulta: 13 junio 2017]. [Online]. Available: <http://www.neurobs.com/> 13
- [20] J. Peirce, “Psychopy,” [consulta: 13 junio 2017]. [Online]. Available: <http://www.psychopy.org/> 13
- [21] M. Kleiner, “Psychtoolbox,” [consulta: 13 junio 2017]. [Online]. Available: <http://psychtoolbox.org/> 13
- [22] A. D. Straw, “Vission egg,” [consulta: 13 junio 2017]. [Online]. Available: <https://visionegg.readthedocs.io/en/latest/index.html> 13
- [23] I. S. MacKenzie, “Evaluating eye tracking systems for computer input,” in *Gaze Interaction and Applications of Eye Tracking: Advances in Assistive Technologies*, P. Majaranta [et al.], Ed. Hershey, PA: IGI Global, 2011, ch. 15, pp. 205–225. 16, 17
- [24] J. Pierce, “Generating stimuli for neuroscience using psychopy,” *Frontiers in Neuroinformatics*, vol. 2, p. 10, 2009. 18
- [25] S. Simpson, “iohub,” [consulta: 23 agosto 2017]. [Online]. Available: <http://www.isolver-solutions.com/iohubdocs/index.html> 22



# Apéndice A

## Configuración del entorno de programación

Entorno de anaconda utilizado.

```
1 name: Memoria
2 channels:
3   - menpo
4   - jlaforet
5   - cogsci
6   - conda-forge
7   - defaults
8 dependencies:
9   - pygame=1.9.2a0
10  - pyglet=1.2.4
11  - alabaster=0.7.10
12  - astroid=1.5.3
13  - babel=2.5.1
14  - backports=1.0
15  - backports.functools_lru_cache=1.4
16  - backports.shutil_get_terminal_size=1.0.0
17  - backports_abc=0.5
18  - bleach=2.0.0
19  - bzip2=1.0.6
20  - ca-certificates=2017.11.5
21  - chardet=3.0.4
22  - colorama=0.3.9
23  - configparser=3.5.0
24  - cycler=0.10.0
25  - cython=0.24.1
26  - decorator=4.1.2
27  - docutils=0.14
28  - entrypoints=0.2.3
29  - enum34=1.1.6
30  - et_xmlfile=1.0.1
31  - freetype=2.6.3
32  - functools32=3.2.3.2
33  - future=0.16.0
34  - h5py=2.6.0
35  - html5lib=1.0.1
36  - idna=2.6
37  - imagesize=0.7.1
38  - ipaddress=1.0.18
```

```

39 - ipykernel=4.7.0
40 - ipython=5.5.0
41 - ipython_genutils=0.2.0
42 - isort=4.2.15
43 - jinja2=2.10
44 - jpeg=9b
45 - jsonschema=2.5.1
46 - jupyter_client=5.2.1
47 - jupyter_core=4.4.0
48 - lazy-object-proxy=1.3.1
49 - libiconv=1.15
50 - libpng=1.6.34
51 - libtiff=4.0.6
52 - libwebp=0.5.2
53 - libxml2=2.9.3
54 - markupsafe=1.0
55 - matplotlib=2.0.0
56 - mccabe=0.6.1
57 - mistune=0.8.3
58 - nbconvert=5.3.1
59 - nbformat=4.4.0
60 - ndg-httpsclient=0.4.2
61 - nibabel=2.0.2
62 - numexpr=2.6.4
63 - numpydoc=0.7.0
64 - opencv=3.2.0
65 - openssl=1.0.2n
66 - pandas=0.18.1
67 - pandoc=2.1.1
68 - pandocfilters=1.4.1
69 - path.py=10.3.1
70 - pathlib2=2.1.0
71 - pep8=1.7.1
72 - pillow=3.2.0
73 - pip=9.0.1
74 - prompt_toolkit=1.0.15
75 - py=1.4.31
76 - pyasn1=0.1.9
77 - pycparser=2.14
78 - pydicom=0.9.9
79 - pyflakes=1.6.0
80 - pygments=2.1.3
81 - pylint=1.8.1
82 - pyopengl=3.1.1a1
83 - pyopenssl=16.2.0
84 - pyosf=1.0.4
85 - pyparsing=2.2.0
86 - pyqt=4.11.4
87 - pyserial=3.4
88 - pytest=2.9.2
89 - python=2.7.14
90 - python-dateutil=2.6.1
91 - pytz=2017.3
92 - pyxid=1.0
93 - pyyaml=3.11
94 - qt=4.8.7
95 - qtawesome=0.4.4
96 - qtconsole=4.3.1
97 - qtpy=1.3.1
98 - requests=2.10.0
99 - setuptools=38.4.0
100 - simplegeneric=0.8.1
101 - singledispatch=3.4.0.3
102 - sip=4.18
103 - six=1.11.0
104 - snowballstemmer=1.2.1
105 - sphinx_rtd_theme=0.2.4
106 - spyder=3.1.4

```

```

107 - ssl_match_hostname=3.5.0.1
108 - subprocess32=3.2.7
109 - testpath=0.3.1
110 - tornado=4.5.3
111 - tqdm=4.7.2
112 - traitlets=4.2.1
113 - vc=9
114 - vs2008_runtime=9.0.30729.6161
115 - vs2015_runtime=14.0.25420
116 - wcwidth=0.1.7
117 - webencodings=0.5
118 - wheel=0.30.0
119 - win_unicode_console=0.5
120 - wincertstore=0.2
121 - wrapt=1.10.11
122 - yaml=0.1.7
123 - zlib=1.2.11
124 - certifi=2017.11.5
125 - cffi=1.6.0
126 - comtypes=0.6.2
127 - configobj=5.0.6
128 - cryptography=1.4
129 - event=1.1.1
130 - greenlet=0.4.9
131 - hdf5=1.8.15.1
132 - icc_rt=2017.0.4
133 - intel-openmp=2018.0.0
134 - jedi=0.8.1
135 - mkl=11.3.3
136 - msgpack-python=0.4.7
137 - numpy=1.11.2
138 - pickleshare=0.7.2
139 - psutil=4.2.0
140 - pytables=3.2.2
141 - pyzmq=15.2.0
142 - rope=0.9.4
143 - scipy=0.17.1
144 - seaborn=0.7.0
145 - sphinx=1.4.1
146 - vs2010_runtime=10.00.40219.1
147 - wxpython=3.0
148 - xlrd=0.9.4
149 - imageio=1.5
150 - moviepy=0.2.2.11
151 - ffmpeg=2.7.0
152 - pip:
153 - backports.ssl-match-hostname==3.5.0.1
154 - egi==0.9.0
155 - glumpy==1.0.6
156 - h5pyviewer==0.0.1.6
157 - hid==0.1.1
158 - jdcal==1.3
159 - openpyxl==2.3.5
160 - psychopy==1.84.2
161 - psychopy-ext==0.6.0.4
162 - pyfilesec==0.2.14
163 - pyparallel==0.2.2
164 - pypiwin32==219
165 - pysoundcard==0.5.2
166 - pysoundfile==0.8.1
167 - python-bidi==0.4.0
168 - svgwrite==1.1.7
169 - tables==3.2.2
170 - triangle==20170429

```

# Apéndice B

## Código fuente

Código para el manejo de datos.

```
1  # -*- coding: utf-8 -*-
2  # =====
3  # Modules
4  # =====
5  import os
6  import copy
7  import numpy as np
8  import sqlite3 as lite
9  from psychopy import visual, colors
10
11
12  # =====
13  # Class: Utils
14  # =====
15  class Utils(object):
16      # =====
17      def __init__(self):
18          pass
19
20      # =====
21      @staticmethod
22      def format_text(word, lmin=0, lmax=-1):
23          try:
24              temp = unicode(word)
25              if lmax <= lmin <= len(temp) or lmin <= len(temp) <= lmax:
26                  return temp
27              else:
28                  return u''
29          except ValueError:
30              return u''
31
32      @staticmethod
33      def format_int(value, vmin=0, default=None):
34          try:
35              temp = int(value)
36              if temp >= vmin:
37                  return temp
38              else:
39                  return default
40          except ValueError:
41              return default
42
```

```

43     @staticmethod
44     def format_float(value, vmin=0.0, default=None):
45         try:
46             temp = float(value)
47             if temp >= vmin:
48                 return temp
49             else:
50                 return default
51         except ValueError:
52             return default
53
54     @staticmethod
55     def format_bool(state, default=False):
56         try:
57             temp = bool(state)
58             return temp
59         except ValueError:
60             return default
61
62     @staticmethod
63     def format_path(path):
64         import platform
65         # -----
66         path = Utils.format_text(path)
67         is_win = any(platform.win32_ver())
68         # -----
69         path = path.replace(u'\\', u'#').replace(u'/', u'#')
70         return path.replace(u'#', u'\\') if is_win else path.replace(u'#', u'/')
71
72     @staticmethod
73     def get_time(date):
74         import pytz
75         from datetime import datetime as dt
76         # -----
77         try:
78             gmt0 = pytz.timezone(u"GMT+0")
79             cltc = pytz.timezone(u"Chile/Continental")
80             date = Utils.format_text(date)
81             date = dt.strptime(date, u'%Y-%m-%d %H:%M:%S')
82             date = date.replace(tzinfo=gmt0)
83             return unicode(date.astimezone(cltc).strftime(u'%Y-%m-%d %H:%M:%S'))
84         except ValueError:
85             return u'No disponible'
86
87
88 # =====
89 # Class: SaccadeDB
90 # =====
91 class SaccadeDB(object):
92     # =====
93     def __init__(self, filepath=u'saccadedb.sqlite3'):
94         self.__db_connection = None
95         self.__db_file = filepath
96         self.__db_script = self.__get_script_path()
97         # -----
98         self.connect()
99
100 # =====
101     @staticmethod
102     def __get_script_path():
103         path_base = os.path.split(os.path.realpath(__file__))[0]
104         path_conf = Utils.format_path(u'/resources/database/')
105         return path_base + path_conf + u'saccadedb.sqlite3'
106
107 # =====
108     def connect(self):
109         print u"Connecting to DB... "
110         if os.path.isfile(self.__db_file):

```

```

111         self.__db_connection = lite.connect(self.__db_file)
112         self.__db_connection.executescript(u"pragma recursive_triggers=1; pragma
foreign_keys=1;")
113         print u'Connected!'
114     else:
115         sql = open(self.__db_script, u'r').read()
116         self.__db_connection = lite.connect(self.__db_file)
117         self.__db_connection.executescript(sql)
118         print u"Database not found. A new one was created."
119
120     def close(self):
121         try:
122             self.__db_connection.close()
123             print u"Disconnected!"
124             return True
125         except lite.Error, event:
126             print u"Error: %s" % event.args[0]
127             return False
128
129     # =====
130     def push_query(self, query):          # insert, update, delete
131         try:
132             self.__db_connection.executescript(query)
133             self.__db_connection.commit()
134             return True
135         except lite.Error, event:
136             if self.__db_connection:
137                 self.__db_connection.rollback()
138             print u"Error: %s" % event.args[0]
139             return False
140
141     def pull_query(self, query):          # select
142         try:
143             cursor = self.__db_connection.cursor()
144             cursor.execute(query)
145             result = cursor.fetchall()
146             result = np.array(result)
147             if result.shape[0] > 0:
148                 return result
149             else:
150                 return None
151         except lite.Error, event:
152             print u"Error: %s" % event.args[0]
153             return None
154
155     # =====
156     # Class: Utils
157     # =====
158     class Master(object):
159     def __init__(self):
160         self.__in_db = False
161         self.__database = None
162         # -----
163         self.__name = u'Unnamed'
164         self.__screen = 0
165         self.__tracker = u'none'
166         self.__monitor = u'default'
167         self.__path = self.__get_base_path()
168
169     # =====
170     @classmethod
171     def get_list(cls, db):
172         if isinstance(db, SaccadeDB):
173             sql = u"""
174             select mas_name
175             from master
176             order by mas_name asc;

```

```

178         """
179         return db.pull_query(query=sql)
180     else:
181         return None
182
183     # =====
184     @staticmethod
185     def get_available_trackers():
186         import glob as gl
187         # -----
188         path_base = os.path.split(os.path.realpath(__file__))[0]
189         path_conf = Utils.format_path(u'/resources/eyetrackers/')
190         path_full = path_base + path_conf
191         # -----
192         return [os.path.basename(item).replace(u'__config.yaml', u'') for item in gl.glob(
193             path_full+u'*.yaml')]
194
195     @staticmethod
196     def get_available_screens():
197         import pygame
198         # -----
199         display = pygame.window.Display()
200         screens = display.get_screens()
201         # -----
202         scr_num = 1
203         scr_lst = []
204         for screen in screens:
205             scr_lst.append(u"monitor %d: (w=%s, h=%s)" % (scr_num, screen.width, screen.
206                 height))
207             scr_num += 1
208         # -----
209         return scr_lst
210
211     @staticmethod
212     def get_available_monitors():
213         from psychopy import monitors
214         # -----
215         return monitors.getAllMonitors()
216
217     @staticmethod
218     def __get_base_path():
219         import sys
220         # -----
221         path_base = os.path.dirname(os.path.realpath(sys.argv[0]))
222         path_fold = Utils.format_path(u'/events')
223         return path_base + path_fold
224
225     # =====
226     def set_database(self, db):
227         if isinstance(db, SaccadeDB):
228             self.__database = db
229             return True
230         else:
231             return False
232
233     def get_database(self):
234         return self.__database
235
236     def is_on_database(self):
237         return self.__in_db
238
239     # -----
240     def set_name(self, name):
241         name = Utils.format_text(name, lmin=3, lmax=50)
242         if self.__database is not None and name != u'':
243             sql = u"select * from master where mas_name='%s';" % name
244             mas_res = self.__database.pull_query(query=sql)
245             # -----

```

```

244         if mas_res is None:
245             self.__name = name
246             return True
247         else:
248             return False
249     else:
250         return False
251
252 def get_name(self):
253     return self.__name
254
255 # -----
256 def set_screen(self, screen):
257     screen = Utils.format_int(screen, default=0)
258     if 0 <= screen < len(self.get_available_screens()):
259         self.__screen = screen
260         return True
261     else:
262         return False
263
264 def get_screen(self):
265     try:
266         return self.get_available_screens()[self.__screen]
267     except:
268         return self.get_available_screens()[0]
269
270 # -----
271 def set_monitor(self, monitor):
272     monitor = Utils.format_text(monitor)
273     if monitor in self.get_available_monitors():
274         self.__monitor = monitor
275         return True
276     else:
277         return False
278
279 def get_monitor(self):
280     return self.__monitor
281
282 # -----
283 def set_tracker(self, tracker):
284     tracker = Utils.format_text(tracker)
285     if tracker in self.get_available_trackers():
286         self.__tracker = tracker
287         return True
288     else:
289         return False
290
291 def get_tracker_name(self):
292     return self.__tracker
293
294 def get_tracker_conf_path(self):
295     if self.__tracker in self.get_available_trackers():
296         path_base = os.path.split(os.path.realpath(__file__))[0]
297         path_conf = Utils.format_path(u'/resources/eyetrackers/')
298         return path_base + path_conf + self.__tracker + u'_config.yaml'
299     else:
300         return u''
301
302 # -----
303 def set_experiment_path(self, path):
304     # import sys
305     # -----
306     exp_path = Utils.format_text(path, lmin=0, lmax=200)
307     if os.path.isdir(exp_path):
308         # base_path = os.path.dirname(os.path.realpath(sys.argv[0]))
309         # self.__path = os.path.relpath(exp_path, base_path)
310         self.__path = exp_path
311         return True

```



```

312         else:
313             return False
314
315     def get_experiment_path(self):
316         return self.__path
317
318     # =====
319     def load(self, name):
320         name = Utils.format_text(name, lmin=3, lmax=50)
321         if self.__database is not None and name != u'':
322             sql = u"""
323             select
324             mas_screen, mas_monitor, mas_tracker, mas_path
325             from master
326             where mas_name='%s';
327             """ % name
328             mas_res = self.__database.pull_query(query=sql)
329             # -----
330             if mas_res is not None:
331                 self.__in_db = True
332                 self.__name = name
333                 print u"Configuration profile %s loaded." % name
334                 # -----
335                 self.__screen = int(mas_res[0, 0])
336                 self.__monitor = unicode(mas_res[0, 1])
337                 self.__tracker = unicode(mas_res[0, 2])
338                 self.__path = unicode(mas_res[0, 3])
339                 # -----
340                 return True
341             else:
342                 print u"Configuration profile %s doesn't exists." % name
343                 return False
344         else:
345             print u"Error: Database or configuration profile name not configured!"
346             return False
347
348     def save(self):
349         if self.__database is not None and self.__name != u'' and self.__tracker != u'none':
350             sql = u"""
351             insert or replace into master
352             (mas_name, mas_screen, mas_tracker, mas_monitor, mas_path)
353             values ('%s', '%d', '%s', '%s', '%s')
354             """ % (self.__name, self.__screen, self.__tracker, self.__monitor, self.__path)
355             mas_res = self.__database.push_query(query=sql)
356             # -----
357             if mas_res:
358                 print u"Configuration profile %s saved." % self.__name
359             else:
360                 print u"Configuration profile %s not saved." % self.__name
361             self.__in_db = mas_res
362             return mas_res
363         else:
364             print u"Error: Database or configuration profile identifiers not configured!"
365             return False
366
367     def copy(self, name):
368         new_mas = copy.deepcopy(self)
369         # -----
370         new_mas.set_database(db=self.__database)
371         new_mas.__in_db = False
372         # -----
373         name_check = new_mas.set_name(name=name)
374         # -----
375         if name_check:
376             return new_mas
377         else:
378             return None
379

```

```

380 def remove(self):
381     if self.__in_db:
382         sql = u"delete from master where mas_name='%s';" % self.__name
383         mas_res = self.__database.push_query(query=sql)
384         # -----
385         self.__in_db = not mas_res
386         return mas_res
387     else:
388         return False
389
390 # =====
391 def get_iohub(self):
392     import yaml
393     # -----
394     if self.__in_db:
395         configuration = {
396             u'monitor_devices': [
397                 {u'Display': {
398                     u'name': u'display',
399                     u'reporting_unit_type': u'pix',
400                     u'device_number': self.__screen,
401                     u'psychopy_monitor_name': self.__monitor,
402                     u'override_using_pscho_settings': True,
403                 }},
404                 {u'Keyboard': {
405                     u'name': u'keyboard',
406                     u'enable': True,
407                     u'save_events': True,
408                 }},
409                 {u'Experiment': {
410                     u'name': u'experimentRuntime',
411                     u'enable': True,
412                     u'save_events': True,
413                 }}
414             ],
415             u'data_store': {
416                 u'enable': True,
417             }
418         }
419         # -----
420         tracker_path = self.get_tracker_conf_path()
421         tracker = yaml.load(open(tracker_path, u'r'))[u'monitor_devices'][0]
422         configuration[u'monitor_devices'].append(tracker)
423         # -----
424         return configuration
425     else:
426         return None
427
428 def get_configuration(self):
429     if self.__in_db:
430         configuration = {
431             u'name': self.__name,
432             u'screen': self.__screen,
433             u'tracker': self.__tracker,
434             u'monitor': self.__monitor,
435             u'experiment_path': self.__path,
436         }
437         return configuration
438     else:
439         return None
440
441 # =====
442 # Class type: ItemList
443 # =====
444 class ItemList(object):
445     # =====
446     def __init__(self, itemclass):

```

```

448         self.__item_class = itemclass
449         self.__item_array = None
450
451     # =====
452     def _item_add(self, item):
453         if isinstance(item, self.__item_class):
454             itm_num = self._item_number()
455             if itm_num is None:
456                 self.__item_array = np.array([item], dtype=self.__item_class)
457             else:
458                 self.__item_array = np.insert(arr=self.__item_array, obj=itm_num, values=[
459 item], axis=0)
460             return True
461         else:
462             return False
463
464     def _item_copy(self, index):
465         itm_num = self._item_number()
466         if itm_num is not None and 0 <= index < itm_num:
467             new_itm = self.__item_array[index].copy()
468             return self._item_add(item=new_itm)
469         else:
470             return False
471
472     def _item_delete(self, index):
473         itm_num = self._item_number()
474         if itm_num is not None and 0 <= index < itm_num:
475             self.__item_array = np.delete(arr=self.__item_array, obj=index, axis=0) if
476 itm_num > 1 else None
477             return True
478         else:
479             return False
480
481     def _item_move_up(self, index):
482         itm_num = self._item_number()
483         if itm_num is not None and 0 < index < itm_num:
484             temp = self.__item_array[index - 1]
485             self.__item_array[index - 1] = self.__item_array[index]
486             self.__item_array[index] = temp
487             return True
488         else:
489             return False
490
491     def _item_move_down(self, index):
492         itm_num = self._item_number()
493         if itm_num is not None and 0 <= index < itm_num-1:
494             temp = self.__item_array[index + 1]
495             self.__item_array[index + 1] = self.__item_array[index]
496             self.__item_array[index] = temp
497             return True
498         else:
499             return False
500
501     def _item_get_all(self):
502         return self.__item_array
503
504     def _item_get_by_index(self, index):
505         itm_num = self._item_number()
506         if itm_num is not None and 0 <= index < itm_num:
507             return self.__item_array[index]
508         else:
509             return None
510
511     def _item_number(self):
512         if self.__item_array is not None:
513             return len(self.__item_array)
514         else:
515             return None

```

```

514
515
516 # =====
517 # Class: Component
518 # =====
519 class Component(object):
520     # =====
521     def __init__(self):
522         self.__name = u'Unnamed'
523         self.__units = u'deg'
524         self.__pos = (0.0, 0.0)
525         self.__ori = 0.0
526         self.__size = 1.0
527         # -----
528         self.__image = None
529         self.__shape = u'square'
530         self.__color = u'white'
531
532     # =====
533     @classmethod
534     def get_list(cls, db, exp, tes, fra):
535         sql = u"""
536         select com_index, com_name, com_shape
537         from component
538         where exp_code='%s' and tes_index='%d' and fra_index='%d'
539         order by com_index asc;
540         """ % (exp, tes, fra)
541         return db.pull_query(query=sql)
542
543     # =====
544     def set_name(self, name):
545         name = Utils.format_text(name, lmin=3, lmax=50)
546         if name != u'':
547             self.__name = name
548             return True
549         else:
550             return False
551
552     def get_name(self):
553         return self.__name
554
555     # -----
556     def set_units(self, units):
557         units = Utils.format_text(units, lmin=2, lmax=20)
558         if units in [u'norm', u'cm', u'deg', u'degFlat', u'degFlatPos', u'pix']:
559             self.__units = units
560             return True
561         else:
562             return False
563
564     def get_units(self):
565         return self.__units
566
567     # -----
568     def set_position(self, posx, posy):
569         posx = Utils.format_float(posx)
570         posy = Utils.format_float(posy)
571         if posx is not None and posy is not None:
572             self.__pos = (posx, posy)
573             return True
574         else:
575             return False
576
577     def get_position(self):
578         return self.__pos
579
580     # -----
581     def set_orientation(self, ori):

```

```

582         ori = Utils.format_float(ori)
583         if ori is not None:
584             self.__ori = ori
585             return True
586         else:
587             return False
588
589     def get_orientation(self):
590         return self.__ori
591
592     # -----
593     def set_size(self, size):
594         size = Utils.format_float(size)
595         if size is not None:
596             self.__size = size
597             return True
598         else:
599             return False
600
601     def get_size(self):
602         return self.__size
603
604     # -----
605     def set_image(self, imagepath):
606         from PIL import Image
607         # -----
608         imagepath = Utils.format_text(imagepath)
609         if imagepath is not u'' and os.path.isfile(imagepath):
610             self.__shape = u'image'
611             self.__image = Image.open(imagepath)
612             return True
613         else:
614             return False
615
616     def get_image(self):
617         return self.__image
618
619     # -----
620     def set_shape(self, shape):
621         shape = Utils.format_text(shape, lmin=5, lmax=20)
622         if shape in [u'arrow', u'circle', u'cross', u'gauss', u'square']:
623             self.__shape = shape
624             return True
625         else:
626             return False
627
628     def get_shape(self):
629         return self.__shape
630
631     # -----
632     def set_color(self, color):
633         color = Utils.format_text(color, lmin=3, lmax=20)
634         if colors.isValidColor(color):
635             self.__color = color
636             return True
637         else:
638             return False
639
640     def get_color(self):
641         return self.__color
642
643     # =====
644     def __decode_image(self, encimg):
645         from PIL import Image
646         from io import BytesIO
647         # -----
648         coded = Utils.format_text(encimg)
649         if coded != u'':

```

```

650         img_buff = BytesIO()
651         img_buff.write(coded.decode(u'base64'))
652         # -----
653         self.__shape = u'image'
654         self.__image = Image.open(img_buff)
655     else:
656         self.__image = None
657
658     def __encode_image(self):
659         from io import BytesIO
660         # -----
661         if self.__image is not None:
662             img_buff = BytesIO()
663             self.__image.save(img_buff, u'PNG')
664             return img_buff.getvalue().encode(u'base64')
665         else:
666             return u''
667
668     # =====
669     def load(self, db, exp, tes, fra, com):
670         sql = u"""
671         select
672         com_name, com_units, com_pos_x, com_pos_y, com_orientation, com_size, com_image,
673         com_shape, com_color
674         from component
675         where exp_code='%s' and tes_index='%d' and fra_index='%d' and com_index='%d';
676         """ % (exp, tes, fra, com)
677         com_res = db.pull_query(query=sql)
678         # -----
679         if com_res is not None:
680             print u"Exp %s, Tes %d, Fra %d: Component %d loaded." % (exp, tes, fra, com)
681             # -----
682             self.__name = unicode(com_res[0, 0])
683             self.__units = unicode(com_res[0, 1])
684             self.__pos = (float(com_res[0, 2]),
685                           float(com_res[0, 3]))
686             self.__ori = float(com_res[0, 4])
687             self.__size = float(com_res[0, 5])
688             self.__shape = unicode(com_res[0, 7])
689             self.__color = unicode(com_res[0, 8])
690             # -----
691             self.__decode_image(unicode(com_res[0, 6]))
692             # -----
693             return True
694         else:
695             print u"Exp %s, Tes %d, Fra %d: Component %d doesn't exists." % (exp, tes, fra,
696             com)
697             return False
698
699     def save(self, db, exp, tes, fra, com):
700         sql = u"""
701         insert into component
702         (exp_code, tes_index, fra_index, com_index,
703         com_name, com_units, com_pos_x, com_pos_y, com_orientation, com_size,
704         com_image, com_shape, com_color)
705         values ('%s', '%d', '%d', '%d', '%s', '%s', '%f', '%f', '%f', '%f', '%s', '%s', '%s
706         ');
707         """ % (
708             exp, tes, fra, com,
709             self.__name, self.__units, self.__pos[0], self.__pos[1], self.__ori, self.__size
710             ,
711             self.__encode_image(), self.__shape, self.__color
712             )
713         com_res = db.push_query(query=sql)
714         # -----
715         if com_res:
716             print u"Exp %s, Tes %d, Fra %d: Component %d saved." % (exp, tes, fra, com)
717         else:

```

```

714         print u"Exp %s, Tes %d, Fra %d: Component %d not saved." % (exp, tes, fra, com)
715     # -----
716     return com_res
717
718     def copy(self):
719         return copy.deepcopy(self)
720
721     # =====
722     def get_execution(self, win):
723         if isinstance(win, visual.Window):
724             if self.__shape == u'image':
725                 return visual.ImageStim(win=win, name=self.__name, image=self.__image,
726                                         pos=self.__pos, ori=self.__ori, units=self.__units)
727             elif self.__shape == u'arrow':
728                 return visual.ShapeStim(win=win, name=self.__name, lineColor=self.__color,
729                                         fillColor=self.__color,
730                                         size=self.__size, pos=self.__pos, ori=self.__ori,
731                                         vertices=((1.0, 0.0), (0.6667, 0.1667), (0.6667,
732                                         (0.0, -0.0667), (0.6667, -0.0667),
733                                         (0.6667, -0.1667)))
734             else:
735                 return visual.GratingStim(win=win, name=self.__name, color=self.__color, sf=
736                 0,
737                 mask=None if self.__shape == u'square' else self.
738                 __shape,
739                 size=self.__size, pos=self.__pos, ori=self.__ori,
740                 units=self.__units)
741             else:
742                 print u"Error: 'win' must be a psychopy visual.Window instance."
743                 return None
744
745     def get_configuration(self):
746         component = {
747             u'name': self.__name,
748             u'units': self.__units,
749             u'position': self.__pos,
750             u'orientation': self.__ori,
751             u'size': self.__size,
752             u'image': self.__encode_image(),
753             u'shape': self.__shape,
754             u'color': self.__color
755         }
756         return component
757
758     # =====
759     # Class: Frame (child of ItemList)
760     # =====
761     class Frame(ItemList):
762         # =====
763         def __init__(self):
764             super(Frame, self).__init__(itemclass=Component)
765             # -----
766             self.__name = u'Unnamed'
767             self.__color = u'black'
768             self.__is_task = False
769             self.__keys_allowed = u''
770             self.__keys_selected = u''
771             self.__time = 0.5
772
773         # =====
774         @classmethod
775         def get_list(cls, db, exp, tes):
776             sql = u"""
777             select fra_index, fra_name
778             from frame

```

```

775         where exp_code='%s' and tes_index='%d'
776         order by fra_index asc;
777         """ % (exp, tes)
778         return db.pull_query(query=sql)
779
780     # =====
781     def set_name(self, name):
782         name = Utils.format_text(name, lmin=3, lmax=20)
783         if name != u'':
784             self.__name = name
785             return True
786         else:
787             return False
788
789     def get_name(self):
790         return self.__name
791
792     # -----
793     def set_color(self, color):
794         color = Utils.format_text(color, lmin=3, lmax=20)
795         if colors.isValidColor(color):
796             self.__color = color
797             return True
798         else:
799             return False
800
801     def get_color(self):
802         return self.__color
803
804     # -----
805     def set_as_task(self, state):
806         self.__is_task = Utils.format_bool(state, default=self.__is_task)
807         if self.__is_task:
808             self.__time = 0.0
809             return True
810         else:
811             self.__keys_allowed = u''
812             self.__keys_selected = u''
813             return False
814
815     def get_state(self):
816         return self.__is_task
817
818     # -----
819     def set_time(self, value):
820         value = Utils.format_float(value)
821         if not self.__is_task and value is not None:
822             self.__time = value
823             return True
824         else:
825             self.__time = 0.0
826             return False
827
828     def get_time(self):
829         return self.__time
830
831     # -----
832     def set_keys_allowed(self, keys):
833         keys = Utils.format_text(keys).replace(unicode(u' '), unicode(u' '))
834         if self.__is_task and keys != u'':
835             self.__keys_allowed = keys
836             return True
837         else:
838             self.__keys_allowed = u''
839             return False
840
841     def get_keys_allowed(self):
842         return self.__keys_allowed

```



```

843
844 # -----
845 def set_keys_selected(self, keys):
846     keys = Utils.format_text(keys).replace(unicode(u' '), unicode(u' '))
847     keys.replace(u" ", u" ")
848     if self.__is_task and self.__keys_allowed != u'' and keys != u'':
849         keys_alw = self.__keys_allowed.split(u',')
850         keys_sel = keys.split(u',')
851         # -----
852         match = [key for key in keys_sel if key in keys_alw]
853         if len(match) == len(keys_sel):
854             self.__keys_selected = keys
855             return True
856         else:
857             self.__keys_selected = u''
858             return True
859     else:
860         self.__keys_selected = u''
861         return False
862
863 def get_keys_selected(self):
864     return self.__keys_selected
865
866 # =====
867 def component_add(self, item):
868     return self._item_add(item=item)
869
870 def component_copy(self, index):
871     return self._item_copy(index=index)
872
873 def component_delete(self, index):
874     return self._item_delete(index=index)
875
876 def component_move_up(self, index):
877     return self._item_move_up(index=index)
878
879 def component_move_down(self, index):
880     return self._item_move_down(index=index)
881
882 def component_get_by_index(self, index):
883     return self._item_get_by_index(index=index)
884
885 def component_get_all(self):
886     return self._item_get_all()
887
888 def component_number(self):
889     return self._item_number()
890
891 # =====
892 def load(self, db, exp, tes, fra):
893     sql = u"""
894     select
895     fra_name, fra_color, fra_is_task, fra_time, fra_keys_allowed, fra_keys_selected
896     from frame
897     where exp_code='%s' and tes_index='%d' and fra_index='%d';
898     """ % (exp, tes, fra)
899     fra_res = db.pull_query(query=sql)
900     # -----
901     if fra_res is not None:
902         print u"Exp %s, Tes %d: Frame %d loaded." % (exp, tes, fra)
903         # -----
904         self.__name = unicode(fra_res[0, 0])
905         self.__color = unicode(fra_res[0, 1])
906         self.__is_task = bool(int(fra_res[0, 2]))
907         self.__time = float(fra_res[0, 3])
908         self.__keys_allowed = unicode(fra_res[0, 4])
909         self.__keys_selected = unicode(fra_res[0, 5])
910         # -----

```

```

911         self.__load_components(db=db, exp=exp, tes=tes, fra=fra)
912         # -----
913         return True
914     else:
915         print u"Exp %s, Tes %d: Frame %d doesn't exists." % (exp, tes, fra)
916         return False
917
918     def save(self, db, exp, tes, fra):
919         sql = u"""
920         insert into frame
921         (exp_code, tes_index, fra_index,
922         fra_name, fra_color, fra_is_task, fra_time, fra_keys_allowed, fra_keys_selected)
923         values ('%s', '%d', '%d', '%s', '%s', '%x', '%f', '%s', '%s');
924         """ % (
925             exp, tes, fra,
926             self.__name, self.__color, self.__is_task, self.__time, self.__keys_allowed,
927             self.__keys_selected
928         )
929         fra_res = db.push_query(query=sql)
930         # -----
931         if fra_res:
932             print u"Exp %s, Tes %d: Frame %d saved. Saving components..." % (exp, tes, fra)
933             self.__save_components(db=db, exp=exp, tes=tes, fra=fra)
934         else:
935             print u"Exp %s, Tes %d: Frame %d not saved." % (exp, tes, fra)
936         # -----
937         return fra_res
938
939     def copy(self):
940         return copy.deepcopy(self)
941
942     # =====
943     def __load_components(self, db, exp, tes, fra):
944         com_lst = Component.get_list(db=db, exp=exp, tes=tes, fra=fra)
945         if com_lst is not None:
946             for com in com_lst:
947                 new_com = Component()
948                 new_com.load(db=db, exp=exp, tes=int(tes), fra=int(fra), com=int(com[0]))
949                 self.component_add(item=new_com)
950         else:
951             print u"Exp %s, Tes %d: Frame %d don't have any component saved on the DB." % (
952                 exp, tes, fra)
953
954     def __save_components(self, db, exp, tes, fra):
955         com_num = self.component_number()
956         if com_num is not None:
957             for index in range(com_num):
958                 self.component_get_by_index(index=index).save(db=db, exp=exp, tes=tes, fra=
959                 fra, com=index)
960         else:
961             print u"Exp %s, Tes %d: Frame %d don't have any component to be saved." % (exp,
962             tes, fra)
963
964     # =====
965     def get_execution(self, win):
966         if isinstance(win, visual.Window):
967             com_num = self.component_number()
968             if com_num is not None:
969                 components = [component.get_execution(win=win) for component in self.
970                 component_get_all()]
971             else:
972                 components = None
973         # -----
974         back = visual.Rect(win=win, width=win.size[0], height=win.size[1], units=u'pix',
975             lineColor=self.__color, fillColor=self.__color)
976         # -----
977         frame = {
978             u'is_task': self.__is_task,

```

```

974         u'time':                self.__time,
975         u'allowed_keys':        self.__keys_allowed.replace(u' space', u' ').split(u'
, '),
976         u'correct_keys':        self.__keys_selected.replace(u' space', u' ').split(u
', '),
977         u'correct_keys_str':    self.__keys_selected,
978         u'background':          back,
979         u'components':          components
980     }
981     # -----
982     return frame
983 else:
984     print u"Error: 'win' must be a psychopy visual.Window instance."
985     return None
986
987 def get_configuration(self):
988     com_num = self.component_number()
989     if com_num is not None:
990         components = [component.get_configuration() for component in self.
component_get_all()]
991     else:
992         components = None
993     # -----
994     frame = {
995         u'is_task':                self.__is_task,
996         u'time':                self.__time,
997         u'allowed_keys':        self.__keys_allowed,
998         u'correct_keys':        self.__keys_selected,
999         u'background':          self.__color,
1000         u'components':          components,
1001     }
1002     # -----
1003     return frame
1004
1005
1006 # =====
1007 # Class: Test (child of ItemList)
1008 # =====
1009 class Test(ItemList):
1010     # =====
1011     def __init__(self):
1012         super(Test, self).__init__(itemclass=Frame)
1013         # -----
1014         self.__name = u'Unnamed'
1015         self.__description = u''
1016         self.__quantity = 1
1017
1018     # =====
1019     @classmethod
1020     def get_list(cls, db, exp):
1021         sql = u"""
1022         select tes_index, tes_name, tes_quantity
1023         from test
1024         where exp_code=' %s'
1025         order by tes_index asc;
1026         """ % exp
1027         return db.pull_query(query=sql)
1028
1029     # =====
1030     def set_name(self, name):
1031         name = Utils.format_text(name, lmin=3, lmax=50)
1032         if name != u'':
1033             self.__name = name
1034             return True
1035         else:
1036             return False
1037
1038     def get_name(self):

```

```

1039         return self.__name
1040
1041     # -----
1042     def set_description(self, text):
1043         text = Utils.format_text(text, lmin=10)
1044         if text != u'':
1045             self.__description = text
1046             return True
1047         else:
1048             return False
1049
1050     def get_description(self):
1051         return self.__description
1052
1053     # -----
1054     def set_quantity(self, value):
1055         value = Utils.format_int(value, vmin=1)
1056         if value is not None:
1057             self.__quantity = value
1058             return True
1059         else:
1060             return False
1061
1062     def get_quantity(self):
1063         return self.__quantity
1064
1065     # =====
1066     def frame_add(self, item):
1067         return self._item_add(item=item)
1068
1069     def frame_copy(self, index):
1070         return self._item_copy(index=index)
1071
1072     def frame_delete(self, index):
1073         return self._item_delete(index=index)
1074
1075     def frame_move_up(self, index):
1076         return self._item_move_up(index=index)
1077
1078     def frame_move_down(self, index):
1079         return self._item_move_down(index=index)
1080
1081     def frame_get_by_index(self, index):
1082         return self._item_get_by_index(index=index)
1083
1084     def frame_get_all(self):
1085         return self._item_get_all()
1086
1087     def frame_number(self):
1088         return self._item_number()
1089
1090     # =====
1091     def load(self, db, exp, tes):
1092         sql = u"""
1093         select
1094         tes_name, tes_description, tes_quantity
1095         from test
1096         where exp_code=' %s' and tes_index=' %d';
1097         """ % (exp, tes)
1098         tes_res = db.pull_query(query=sql)
1099         # -----
1100         if tes_res is not None:
1101             print u"Exp %s: Test %d loaded." % (exp, tes)
1102             # -----
1103             self.__name = unicode(tes_res[0, 0])
1104             self.__description = unicode(tes_res[0, 1])
1105             self.__quantity = int(tes_res[0, 2])
1106             # -----

```

```

1107         self.__load_frames(db=db, exp=exp, tes=tes)
1108         # -----
1109         return True
1110     else:
1111         print u"Exp %s: Test %d doesn't exists." % (exp, tes)
1112         return False
1113
1114     def save(self, db, exp, tes):
1115         sql = u"""
1116         insert into test
1117         (exp_code, tes_index, tes_name, tes_description, tes_quantity)
1118         values ('%s', '%d', '%s', '%s', '%d');
1119         """ % (
1120             exp, tes,
1121             self.__name, self.__description, self.__quantity
1122         )
1123         tes_res = db.push_query(query=sql)
1124         # -----
1125         if tes_res:
1126             print u"Exp %s: Test %d saved. Saving frames..." % (exp, tes)
1127             self.__save_frames(db=db, exp=exp, tes=tes)
1128         else:
1129             print u"Exp %s: Test %d not saved." % (exp, tes)
1130         # -----
1131         return tes_res
1132
1133     def copy(self):
1134         return copy.deepcopy(self)
1135
1136     # =====
1137     def __load_frames(self, db, exp, tes):
1138         fra_lst = Frame.get_list(db=db, exp=exp, tes=tes)
1139         if fra_lst is not None:
1140             for fra in fra_lst:
1141                 new_fra = Frame()
1142                 new_fra.load(db=db, exp=exp, tes=int(tes), fra=int(fra[0]))
1143                 self.frame_add(item=new_fra)
1144         else:
1145             print u"Exp %s: Test %d don't have any frame saved on the DB." % (exp, tes)
1146
1147     def __save_frames(self, db, exp, tes):
1148         fra_num = self.frame_number()
1149         if fra_num is not None:
1150             for index in range(fra_num):
1151                 self.frame_get_by_index(index=index).save(db=db, exp=exp, tes=tes, fra=index)
1152         else:
1153             print u"Exp %s: Test %d don't have any frame to be saved." % (exp, tes)
1154
1155     # =====
1156     def get_execution(self, win):
1157         if isinstance(win, visual.Window):
1158             fra_num = self.frame_number()
1159             if fra_num is not None:
1160                 frames = [frame.get_execution(win=win) for frame in self.frame_get_all()]
1161                 test = {
1162                     u'name': self.__name,
1163                     u'sequence': np.full(shape=(self.__quantity, 1), fill_value=1, dtype=
1164 int),
1165                     u'frames': frames
1166                 }
1167                 return test
1168             else:
1169                 return None
1170         # -----
1171     else:
1172         print u"Error: 'win' must be a psychopy visual.Window instance."
1173         return None

```

```

1173
1174     def get_configuration(self):
1175         fra_num = self.frame_number()
1176         if fra_num is not None:
1177             frames = [frame.get_configuration() for frame in self.frame_get_all()]
1178             test = {
1179                 u'name': self.__name,
1180                 u'repetitions': self.__quantity,
1181                 u'description': self.__description,
1182                 u'frames': frames,
1183             }
1184             return test
1185         else:
1186             return None
1187
1188
1189 # =====
1190 # Class: Experiment (child of ItemList)
1191 # =====
1192 class Experiment(ItemList):
1193     # =====
1194     def __init__(self):
1195         super(Experiment, self).__init__(itemclass=Test)
1196         # -----
1197         self.__in_db = False
1198         self.__database = None
1199         # -----
1200         self.__code = u''
1201         self.__name = u'Unnamed'
1202         self.__version = u''
1203         self.__description = u''
1204         self.__instructions = u''
1205         self.__comments = u''
1206         # -----
1207         self.__date_created = u'Not available'
1208         self.__date_updated = u'Not available'
1209         # -----
1210         self.__dia_is_active = True
1211         self.__dia_ask_age = True
1212         self.__dia_ask_gender = True
1213         self.__dia_ask_glasses = True
1214         self.__dia_ask_eye_color = True
1215         # -----
1216         self.__con_need_space = False
1217         self.__con_is_random = False
1218         self.__con_is_rest = False
1219         self.__con_rest_time = 0.0
1220         self.__con_rest_period = 0
1221
1222     # =====
1223     @classmethod
1224     def get_experiment_list(cls, db):
1225         sql = u"""
1226         select exp_code, exp_name, exp_version
1227         from experiment
1228         order by exp_name asc;
1229         """
1230         return db.pull_query(query=sql)
1231
1232     # =====
1233     def set_database(self, db):
1234         if isinstance(db, SaccadeDB):
1235             self.__database = db
1236             return True
1237         else:
1238             return False
1239
1240     def get_database(self):

```

```

1241         return self.__database
1242
1243     def is_on_database(self):
1244         return self.__in_db
1245
1246     # -----
1247     def set_code(self, code):
1248         code = Utils.format_text(code, lmin=3, lmax=10)
1249         if self.__database is not None and code != u'':
1250             sql = u"select * from experiment where exp_code='%s';" % code
1251             exp_res = self.__database.pull_query(query=sql)
1252             # -----
1253             if exp_res is None:
1254                 self.__code = code
1255                 return True
1256             else:
1257                 return False
1258         else:
1259             return False
1260
1261     def get_code(self):
1262         return self.__code
1263
1264     # -----
1265     def set_info(self, name, version):
1266         name = Utils.format_text(name, lmin=3, lmax=50)
1267         version = Utils.format_text(version, lmin=3, lmax=10)
1268         if self.__database is not None and name != u'' and version != u'':
1269             sql = u"select * from experiment where exp_name='%s' and exp_version='%s';" % (
1270 name, version)
1271             exp_res = self.__database.pull_query(query=sql)
1272             # -----
1273             if exp_res is None:
1274                 self.__name = name
1275                 self.__version = version
1276                 return True
1277             else:
1278                 return False
1279         else:
1280             return False
1281
1282     def get_name(self):
1283         return self.__name
1284
1285     def get_version(self):
1286         return self.__version
1287
1288     # -----
1289     def set_descripton(self, text):
1290         text = Utils.format_text(text, lmin=10)
1291         if text != u'':
1292             self.__description = text
1293             return True
1294         else:
1295             return False
1296
1297     def get_description(self):
1298         return self.__description
1299
1300     # -----
1301     def set_comments(self, text):
1302         text = Utils.format_text(text, lmin=10)
1303         if text != u'':
1304             self.__comments = text
1305             return True
1306         else:
1307             return False

```

```

1308     def get_comments(self):
1309         return self.__comments
1310
1311     # -----
1312     def set_instruction(self, text):
1313         text = Utils.format_text(text, lmin=10)
1314         if text != u'':
1315             self.__instructions = text
1316             return True
1317         else:
1318             return False
1319
1320     def get_instruction(self):
1321         return self.__instructions
1322
1323     # -----
1324     def set_dialog(self, status, askage, askgender, askglasses, askeyecolor):
1325         self.__dia_is_active = Utils.format_bool(status, default=self.__dia_is_active)
1326         self.__dia_ask_age = Utils.format_bool(askage, default=self.__dia_ask_age)
1327         self.__dia_ask_gender = Utils.format_bool(askgender, default=self.__dia_ask_gender)
1328         self.__dia_ask_glasses = Utils.format_bool(askglasses, default=self.
1329         __dia_ask_glasses)
1330         self.__dia_ask_eye_color = Utils.format_bool(askeyecolor, default=self.
1331         __dia_ask_eye_color)
1332
1333     def is_dialog_active(self):
1334         return self.__dia_is_active
1335
1336     def is_ask_age(self):
1337         return self.__dia_ask_age
1338
1339     def is_ask_gender(self):
1340         return self.__dia_ask_gender
1341
1342     def is_ask_glasses(self):
1343         return self.__dia_ask_glasses
1344
1345     def is_ask_eye_color(self):
1346         return self.__dia_ask_eye_color
1347
1348     # -----
1349     def set_space_start(self, status):
1350         self.__con_need_space = Utils.format_bool(status, default=self.__con_need_space)
1351
1352     def is_space_start(self):
1353         return self.__con_need_space
1354
1355     # -----
1356     def set_random(self, status):
1357         self.__con_is_random = Utils.format_bool(status, default=self.__con_is_random)
1358
1359     def is_random(self):
1360         return self.__con_is_random
1361
1362     # -----
1363     def set_rest_conf(self, status, period, time):
1364         status = Utils.format_bool(status, default=self.__con_is_rest)
1365         period = Utils.format_int(period, default=-0)
1366         time = Utils.format_float(time, default=-0.0)
1367         if status and period > 0 and time > 0.0:
1368             self.__con_is_rest = status
1369             self.__con_rest_period = period
1370             self.__con_rest_time = time
1371             return True
1372         else:
1373             return False
1374
1375     def is_rest(self):

```



```

1374         return self.__con_is_rest
1375
1376     def get_rest_period(self):
1377         return self.__con_rest_period
1378
1379     def get_rest_time(self):
1380         return self.__con_rest_time
1381
1382     # =====
1383     def test_add(self, item):
1384         return self._item_add(item=item)
1385
1386     def test_copy(self, index):
1387         return self._item_copy(index=index)
1388
1389     def test_delete(self, index):
1390         return self._item_delete(index=index)
1391
1392     def test_move_up(self, index):
1393         return self._item_move_up(index=index)
1394
1395     def test_move_down(self, index):
1396         return self._item_move_down(index=index)
1397
1398     def test_get_by_index(self, index):
1399         return self._item_get_by_index(index=index)
1400
1401     def test_get_all(self):
1402         return self._item_get_all()
1403
1404     def test_number(self):
1405         return self._item_number()
1406
1407     # =====
1408     def load(self, code):
1409         code = Utils.format_text(code, lmin=3, lmax=10)
1410         if self.__database is not None and code != u'':
1411             sql = u"""
1412             select
1413             exp.exp_name, exp.exp_version, exp.exp_description, exp.exp_instructions, exp.
exp_comments,
1414             exp.exp_date_creation, exp.exp_date_update,
1415             dia.dia_is_active, dia.dia_ask_age, dia.dia_ask_gender, dia.dia_ask_glasses, dia
.dia_ask_eye_color,
1416             con.con_need_space, con.con_is_random, con.con_is_rest, con.con_rest_period, con
.con_rest_time
1417             from experiment as exp
1418             inner join exp_dia as dia on exp.exp_code=dia.exp_code
1419             inner join exp_con as con on exp.exp_code=con.exp_code
1420             where exp.exp_code='%s';
1421             """ % code
1422             exp_res = self.__database.pull_query(query=sql)
1423             # -----
1424             if exp_res is not None:
1425                 self.__in_db = True
1426                 self.__code = code
1427                 print u"Experiment %s loaded." % self.__code
1428                 # -----
1429                 self.__name = unicode(exp_res[0, 0])
1430                 self.__version = unicode(exp_res[0, 1])
1431                 self.__description = unicode(exp_res[0, 2])
1432                 self.__instructions = unicode(exp_res[0, 4])
1433                 self.__comments = unicode(exp_res[0, 3])
1434                 self.__date_created = Utils.get_time(exp_res[0, 5])
1435                 self.__date_updated = Utils.get_time(exp_res[0, 6])
1436                 self.__dia_is_active = bool(int(exp_res[0, 7]))
1437                 self.__dia_ask_age = bool(int(exp_res[0, 8]))
1438                 self.__dia_ask_gender = bool(int(exp_res[0, 9]))

```

```

1439         self.__dia_ask_glasses = bool(int(exp_res[0, 10]))
1440         self.__dia_ask_eye_color = bool(int(exp_res[0, 11]))
1441         self.__con_need_space = bool(int(exp_res[0, 12]))
1442         self.__con_is_random = bool(int(exp_res[0, 13]))
1443         self.__con_is_rest = bool(int(exp_res[0, 14]))
1444         self.__con_rest_period = int(exp_res[0, 15])
1445         self.__con_rest_time = float(exp_res[0, 16])
1446         # -----
1447         self.__load_tests()
1448         # -----
1449         return True
1450     else:
1451         print u"Experiment %s doesn't exists." % self.__code
1452         return False
1453     else:
1454         print u"Error: Database or experiment code not configured!"
1455         return False
1456
1457     def save(self):
1458         if self.__database is not None and self.__code != u'' and self.__name != u'':
1459             if self.__in_db:
1460                 sql = u"""
1461                 update experiment set
1462                 exp_name='%s', exp_version='%s', exp_description='%s', exp_comments='%s',
exp_instructions='%s'
1463                 where exp_code='%s';
1464                 update exp_dia set
1465                 dia_is_active='%x', dia_ask_age='%x', dia_ask_gender='%x', dia_ask_glasses=
'%x', dia_ask_eye_color='%x'
1466                 where exp_code='%s';
1467                 update exp_con set
1468                 con_need_space='%x', con_is_random='%x', con_is_rest='%x', con_rest_period=
'%d', con_rest_time='%f'
1469                 where exp_code='%s';
1470                 """ % (
1471                 self.__name, self.__version, self.__description, self.__comments, self.
__instructions, self.__code,
1472                 self.__dia_is_active, self.__dia_ask_age, self.__dia_ask_gender, self.
__dia_ask_glasses,
1473                 self.__dia_ask_eye_color, self.__code, self.__con_need_space, self.
__con_is_random,
1474                 self.__con_is_rest, self.__con_rest_period, self.__con_rest_time, self.
__code
1475                 )
1476             else:
1477                 sql = u"""
1478                 insert into experiment
1479                 (exp_code, exp_name, exp_version, exp_description, exp_comments,
exp_instructions)
1480                 values ('%s', '%s', '%s', '%s', '%s', '%s');
1481                 insert into exp_dia
1482                 (exp_code, dia_is_active, dia_ask_age, dia_ask_gender, dia_ask_glasses,
dia_ask_eye_color)
1483                 values ('%s', '%x', '%x', '%x', '%x', '%x');
1484                 insert into exp_con
1485                 (exp_code, con_need_space, con_is_random, con_is_rest, con_rest_period,
con_rest_time)
1486                 values ('%s', '%x', '%x', '%x', '%d', '%f');
1487                 """ % (
1488                 self.__code, self.__name, self.__version, self.__description, self.
__comments, self.__instructions,
1489                 self.__code, self.__dia_is_active, self.__dia_ask_age, self.
__dia_ask_gender,
1490                 self.__dia_ask_glasses, self.__dia_ask_eye_color, self.__code, self.
__con_need_space,
1491                 self.__con_is_random, self.__con_is_rest, self.__con_rest_period, self.
__con_rest_time
1492                 )

```

```

1493         exp_res = self.__database.push_query(query=sql)
1494         # -----
1495         if exp_res:
1496             print u"Experiment %s saved. Saving tests..." % self.__code
1497             self.__save_tests()
1498         else:
1499             print u"Experiment %s not saved." % self.__code
1500             # -----
1501             self.__in_db = exp_res
1502             return exp_res
1503     else:
1504         print u"Error: Database or experiment basic identifiers not configured!"
1505         return False
1506
1507     def copy(self, code, version):
1508         new_exp = copy.deepcopy(self)
1509         # -----
1510         new_exp.set_database(db=self.__database)
1511         new_exp.__in_db = False
1512         # -----
1513         code_check = new_exp.set_code(code=code)
1514         info_check = new_exp.set_info(name=self.__name, version=version)
1515         # -----
1516         if code_check and info_check:
1517             return new_exp
1518         else:
1519             return None
1520
1521     def remove(self):
1522         if self.__in_db:
1523             sql = u"delete from experiment where exp_code='%s';" % self.__code
1524             exp_res = self.__database.push_query(query=sql)
1525             # -----
1526             self.__in_db = not exp_res
1527             return exp_res
1528         else:
1529             return False
1530
1531     # =====
1532     def __load_tests(self):
1533         tes_lst = Test.get_list(db=self.__database, exp=self.__code)
1534         if tes_lst is not None:
1535             for tes in tes_lst:
1536                 new_tes = Test()
1537                 new_tes.load(db=self.__database, exp=self.__code, tes=int(tes[0]))
1538                 self.test_add(item=new_tes)
1539         else:
1540             print u"Experiment %s don't have any test saved on the DB." % self.__code
1541
1542     def __save_tests(self):
1543         sql = u"delete from test where exp_code='%s';" % self.__code
1544         self.__database.push_query(query=sql)
1545         # -----
1546         tes_num = self.test_number()
1547         if tes_num is not None:
1548             for index in range(tes_num):
1549                 self.test_get_by_index(index=index).save(db=self.__database, exp=self.__code
, tes=index)
1550         else:
1551             print u"Experiment %s don't have any test to be saved." % self.__code
1552
1553     # =====
1554     def get_iohub(self, unixstamp):
1555         if self.__in_db:
1556             experiment = {
1557                 u'title': self.__name,
1558                 u'code': self.__code,
1559                 u'version': self.__version,

```

```

1560         u'description': self.__description,
1561         u'display_experiment_dialog': True,
1562         # -----
1563         u'session_defaults': {
1564             u'name': u'Session...',
1565             u'code': unixstamp,
1566             u'comments': self.__comments,
1567         },
1568         u'display_session_dialog': True,
1569         u'session_variable_order': [u'name', u'code', u'comments'],
1570         # -----
1571         u'ioHub': {
1572             u'enable': True,
1573         },
1574     }
1575     if self.__dia_is_active:
1576         experiment[u'session_defaults'][u'user_variables'] = {}
1577         if self.__dia_ask_age:
1578             experiment[u'session_defaults'][u'user_variables'][u'participant_age'] =
1579             u'Unknown'
1580             experiment[u'session_variable_order'].append(u'participant_age')
1581             if self.__dia_ask_gender:
1582                 experiment[u'session_defaults'][u'user_variables'][u'participant_gender']
1583                 = [u'Male', u'Female']
1584                 experiment[u'session_variable_order'].append(u'participant_gender')
1585                 if self.__dia_ask_glasses:
1586                     experiment[u'session_defaults'][u'user_variables'][u'glasses'] = [u'Yes'
1587                     , u'No']
1588                     experiment[u'session_defaults'][u'user_variables'][u'contacts'] = [u'Yes
1589                     ', u'No']
1590                     experiment[u'session_variable_order'].append(u'glasses')
1591                     experiment[u'session_variable_order'].append(u'contacts')
1592                     if self.__dia_ask_eye_color:
1593                         experiment[u'session_defaults'][u'user_variables'][u'eye_color'] = u'
1594                         Unknown'
1595                         experiment[u'session_variable_order'].append(u'eye_color')
1596
1597         return experiment
1598     else:
1599         print u"Error: To execute a experiment you need to ensure that the experiment is
1600         saved on the DB."
1601         return None
1602
1603     def get_execution(self, win):
1604         if isinstance(win, visual.Window) and self.__in_db:
1605             tes_num = self.test_number()
1606             if tes_num is not None:
1607                 test_list = []
1608                 test_data = []
1609                 for index in range(tes_num):
1610                     test = self.test_get_by_index(index=index)
1611                     test = test.get_execution(win=win)
1612                     if test is not None:
1613                         test_list.append(index*test[u'sequence'])
1614                         test_data.append({
1615                             u'name': test[u'name'],
1616                             u'frames': test[u'frames']
1617                         })
1618                 # -----
1619                 test_list = np.concatenate(test_list)
1620                 if self.__con_is_random:
1621                     np.random.shuffle(test_list)
1622                 # -----
1623                 experiment = {
1624                     u'instruction': self.__instructions,
1625                     u'space_start': self.__con_need_space,
1626                     u'rest_active': self.__con_is_rest,
1627                     u'rest_period': self.__con_rest_period,

```

```

1622         u'rest_time':         self.__con_rest_time,
1623         u'test_sequence':      test_list,
1624         u'test_data':          test_data,
1625     }
1626     # -----
1627     return experiment
1628 else:
1629     return None
1630 else:
1631     print u"Error: To execute a experiment you need to ensure that:" \
1632           u"\n\t- win is instance of pyschopy visual.Window." \
1633           u"\n\t- the experiment is saved on the DB."
1634     return None
1635
1636 def get_configuration(self):
1637     if self.__in_db:
1638         tes_num = self.test_number()
1639         if tes_num is not None:
1640             tests = [test.get_configuration() for test in self.test_get_all()]
1641         else:
1642             tests = None
1643         # -----
1644         configuration = {
1645             u'experiment': {
1646                 u'title':         self.__name,
1647                 u'code':          self.__code,
1648                 u'version':       self.__version,
1649                 u'description':    self.__description,
1650                 u'instruction':    self.__instructions,
1651                 u'session_configuration': {
1652                     u'comments':   self.__comments,
1653                     u'space_start': self.__con_need_space,
1654                     u'randomize':  self.__con_is_random,
1655                     u'rest': {
1656                         u'active':  self.__con_is_rest,
1657                         u'period':  self.__con_rest_period,
1658                         u'time':    self.__con_rest_time,
1659                     },
1660                     u'dialog': {
1661                         u'active':   self.__dia_is_active,
1662                         u'ask_age':  self.__dia_ask_age,
1663                         u'ask_gender': self.__dia_ask_gender,
1664                         u'ask_glasses': self.__dia_ask_glasses,
1665                         u'ask_eye_color': self.__dia_ask_eye_color,
1666                     }
1667                 },
1668                 u'tests': tests
1669             }
1670         }
1671         # -----
1672         return configuration
1673     else:
1674         return None

```

## Código para la ejecución del experimento.

```
1  #- coding: utf-8 #-
2  # =====
3  # Modules
4  # =====
5  import os
6  from psychopy import visual, core
7  from psychopy.iohub import (ioHubExperimentRuntime, getCurrentDateTimeString)
8
9
10 # =====
11 # Script Handler
12 # =====
13 class ExperimentHandler(object):
14     # =====
15     def __init__(self):
16         self.__database = None
17         self.__mas_data = None
18         self.__exp_data = None
19         self.__exp_exec = {}
20         self.__exp_path = u''
21         self.__base_path = u''
22         self.__timestamp = u''
23         self.__is_loaded = False
24         self.__is_ready = False
25
26     # =====
27     def load_experiment(self, db, mas, exp):
28         import time
29         from saccadeApp import SaccadeDB, Master, Experiment
30         # -----
31         if not self.__is_loaded:
32             if isinstance(db, SaccadeDB):
33                 self.__mas_data = Master()
34                 self.__mas_data.set_database(db=db)
35                 mas_check = self.__mas_data.load(name=mas)
36                 # -----
37                 self.__exp_data = Experiment()
38                 self.__exp_data.set_database(db=db)
39                 exp_check = self.__exp_data.load(code=exp)
40                 # -----
41                 if mas_check and exp_check:
42                     self.__is_loaded = True
43                     self.__timestamp = unicode(int(time.time()))
44                     # -----
45                     self.__base_path = self.__mas_data.get_experiment_path()
46                     self.__exp_path = self.__base_path + u'\\' + self.__exp_data.get_name()
47                     # -----
48                     return self.__is_loaded
49         else:
50             return self.__is_loaded
51
52     def save_execution_parameters(self):
53         import yaml
54         import codecs
55         # -----
56         if self.__is_loaded and not self.__is_ready:
57             exp_code = self.__exp_data.get_code()
58             exp_version = self.__exp_data.get_version()
59             # -----
60             exp_version_path = self.__exp_path + u'\\' + u'['+exp_version+u'] [' +exp_code+u']
61
62             exp_version_log_path = exp_version_path + u'\\logs'
63             exp_version_cfg_path = exp_version_path + u'\\config'
64             # -----
65             if not os.path.exists(self.__base_path):
```

```

65         os.makedirs(self.__base_path)
66     if not os.path.exists(self.__exp_path):
67         os.makedirs(self.__exp_path)
68     if not os.path.exists(exp_version_path):
69         os.makedirs(exp_version_path)
70     if not os.path.exists(exp_version_log_path):
71         os.makedirs(exp_version_log_path)
72     if not os.path.exists(exp_version_cfg_path):
73         os.makedirs(exp_version_cfg_path)
74     # -----
75     exp_logfile_config = u'['+exp_code+u']['+self.__timestamp+u']
log_experiment_config.yaml'
76     exp_logfile_master = u'['+exp_code+u']['+self.__timestamp+u']log_master_config.
yaml'
77     exp_cfgfile_config = u'['+exp_code+u']['+self.__timestamp+u']experiment_config.
yaml'
78     exp_cfgfile_iohub = u'[' + exp_code + u'][' + self.__timestamp + u']iohub_config
.yaml'
79     # -----
80     mas_config = self.__mas_data.get_iohub()
81     mas_config[u'data_store'][u'filename'] = exp_version_path + u'\\' + u'[' +
exp_code + u']events_data'
82     filepath = exp_version_cfg_path + u'\\' + exp_cfgfile_iohub
83     with codecs.open(filename=filepath, mode=u'w', encoding=u'utf-8') as outfile:
84         yaml.safe_dump(mas_config, outfile, default_flow_style=None, indent=4)
85     # -----
86     mas_log = self.__mas_data.get_configuration()
87     filepath = exp_version_log_path + u'\\' + exp_logfile_master
88     with codecs.open(filename=filepath, mode=u'w', encoding=u'utf-8') as outfile:
89         yaml.safe_dump(mas_log, outfile, default_flow_style=False, indent=4)
90     # -----
91     exp_config = self.__exp_data.get_iohub(unixstamp=self.__timestamp)
92     exp_config[u'ioHub'][u'config'] = exp_version_cfg_path + u'\\' +
exp_cfgfile_iohub
93     filepath = exp_version_cfg_path + u'\\' + exp_cfgfile_config
94     with codecs.open(filename=filepath, mode=u'w', encoding=u'utf-8') as outfile:
95         yaml.safe_dump(exp_config, outfile, default_flow_style=None, indent=4)
96     # -----
97     exp_log = self.__exp_data.get_configuration()
98     filepath = exp_version_log_path + u'\\' + exp_logfile_config
99     with codecs.open(filename=filepath, mode=u'w', encoding=u'utf-8') as outfile:
100         yaml.safe_dump(exp_log, outfile, default_flow_style=False, indent=4)
101     # -----
102     self.__exp_exec[u'experiment_config_path'] = exp_version_cfg_path
103     self.__exp_exec[u'experiment_config_file'] = exp_cfgfile_config
104     self.__exp_exec[u'experiment_data'] = self.__exp_data
105     # -----
106     self.__is_ready = True
107     return self.__is_ready
108 else:
109     return self.__is_loaded
110
111 def execute_experiment(self):
112     if self.__is_loaded and self.__is_ready:
113         runtime = ExperimentRuntime(self.__exp_exec)
114         runtime.start()
115         # -----
116         return True
117     else:
118         return False
119
120
121 # =====
122 # Script
123 # =====
124 class ExperimentRuntime(ioHubExperimentRuntime):
125     def __init__(self, exp_cfg_dict):
126         super(ExperimentRuntime, self).__init__(configFilePath=exp_cfg_dict[u'

```

```

127     experiment_config_path'],
128                                     configFile=exp_cfg_dict[u'
experiment_config_file'])
129     self.__experiment = exp_cfg_dict[u'experiment_data']
130
131 def run(self, *args):
132     # =====
133     # Prepare Hardware
134     # =====
135     try:
136         tracker = self.hub.devices.tracker
137         tracker.runSetupProcedure()
138         tracker.setRecordingState(False)
139     except Exception:
140         from psychopy.iohub.util import MessageDialog
141         md = MessageDialog(title=u"No Eye Tracker Configuration Found",
142                             msg=u"No eyetracker selected/found. Check the"
143                                 u"experiment settings.",
144                             showButtons=MessageDialog.OK_BUTTON,
145                             dialogType=MessageDialog.ERROR_DIALOG,
146                             allowCancel=False, display_index=0)
147         md.show()
148         return 1
149     # -----
150     display = self.hub.devices.display
151     kb = self.hub.devices.keyboard
152     # =====
153     # Get experiment
154     # =====
155     # Prepare Window
156     resolution = display.getPixelResolution()
157     coordinate = display.getCoordinateType()
158     window = visual.Window(size=resolution, monitor=display.getPsychopyMonitorName(),
159                             units=coordinate,
160                             fullscr=True, allowGUI=False, screen=display.getIndex())
161
162     # Get experiment content
163     exp_data = self.__experiment.get_execution(win=window)
164
165     # Show instructions
166     instruction_screen = visual.TextStim(window, text=u'', pos=(0, 0), height=24, color=
u'white',
167                                     alignHoriz=u'center', alignVert=u'center',
168     wrapWidth=window.size[0] * 0.9)
169     instruction_screen.setText(exp_data[u'instruction'] + u"\n\nPress Any Key to Start
Experiment.")
170     instruction_screen.draw()
171     flip_time = window.flip()
172     # -----
173     self.hub.sendMessageEvent(text=u"=== EXPERIMENT START ===", sec_time=flip_time)
174     self.hub.sendMessageEvent(text=u"===== INFO =====")
175     self.hub.sendMessageEvent(text=u>Date: {0}".format(getCurrentDateTimeString
176     ()))
177     self.hub.sendMessageEvent(text=u"Experiment ID: {0}".format(self.hub.experimentID))
178     self.hub.sendMessageEvent(text=u"Session ID: {0}".format(self.hub.
experimentSessionID))
179     self.hub.sendMessageEvent(text=u"Screen (ID, Size, CoordType): ({0}, {1}, {2})".
format(
180     display.getIndex(), display.getPixelResolution(), display.getCoordinateType()))
181     self.hub.sendMessageEvent(text=u"Screen calculated Pixels Per Degree: {0} x, {1} y".
format(
182     *display.getPixelsPerDegree()))
183     self.hub.sendMessageEvent(text=u"===== END INFO =====")
184     self.hub.clearEvents(u'all')
185     # -----
186     kb.waitForPresses()

```



```

185     # =====
186     # Experiment presentation
187     # =====
188     self.hub.sendMessageEvent(text=u"== TESTS SEQUENCE START ==")
189     # -----
190     test_count = 0
191     for test_index in exp_data[u'test_sequence'][:, 0]:
192         test = exp_data[u'test_data'][test_index]
193         # =====
194         if exp_data[u'rest_active'] and test_count > 0 and test_count % exp_data[u'
rest_period'] == 0:
195             instruction_screen.setText(u"Rest time.")
196             instruction_screen.draw()
197             flip_time = window.flip()
198             # -----
199             self.hub.sendMessageEvent(text=u"Rest time started...")
200             core.wait(exp_data[u'rest_time'])
201             self.hub.sendMessageEvent(text=u"Rest time finished.")
202             # -----
203             if exp_data[u'space_start']:
204                 instruction_screen.setText(u"Test: "+test[u'name']+"u"\n\nPress Space to
Start Experiment.")
205                 instruction_screen.draw()
206                 flip_time = window.flip()
207                 # -----
208                 self.hub.sendMessageEvent(text=u"Waiting user input to start Test (ID:{0} ,
Name: {1})".format(
209                     test_index, test[u'name']))
210                 self.hub.clearEvents(u'all')
211                 kb.waitForPresses(keys=[u' ', ])
212                 # =====
213                 timer = core.Clock()
214                 # -----
215                 frames = test[u'frames']
216                 frame_index = 0
217                 frame_buffer_index = -1
218                 frame_total = len(frames)
219                 # -----
220                 state = u'buffer'
221                 frame = None
222                 frame_buffer = None
223                 is_last_frame = False
224                 is_first_frame = True
225                 is_test_finish = False
226                 # -----
227                 self.hub.sendMessageEvent(text=u"Starting Test (ID:{0} , Name: {1})".format(
test_index, test[u'name']))
228                 # -----
229                 tracker.setRecordingState(True)
230                 while not is_test_finish:
231                     with Switch(state) as case:
232                         if case(u'buffer'):
233                             frame_buffer_index += 1
234                             if frame_buffer_index < frame_total:
235                                 self.hub.sendMessageEvent(text=u>Loading Frame (ID: {0})".format
(frame_buffer_index))
236                                 self.hub.sendMessageEvent(text=u>Loading Frame Components...")
237                                 # -----
238                                 frame_buffer = frames[frame_buffer_index]
239                                 frame_buffer[u'background'].draw()
240                                 if frame_buffer[u'components'] is not None:
241                                     for component in frame_buffer[u'components']:
242                                         component.draw()
243                             else:
244                                 self.hub.sendMessageEvent(text=u"No more frames to load...")
245                                 is_last_frame = True
246                                 # -----
247                                 if is_first_frame:

```

```

248         is_first_frame = False
249         state = u'flip'
250     else:
251         state = u'loop'
252
253     elif case(u'flip'):
254         if is_last_frame:
255             state = u'end'
256         else:
257             frame = frame_buffer
258             frame_index = frame_buffer_index
259             # -----
260             self.hub.sendMessageEvent(text=u"Frame Started (ID: {0})".format
(frame_index))
261             # -----
262             flip_time = window.flip()
263             timer.reset()
264             state = u'buffer'
265
266     elif case(u'loop'):
267         if frame[u'is_task']:                                # is a selection
268             pressed = kb.waitForPresses(keys=frame[u'allowed_keys'])
269             key = unicode(pressed[len(pressed)-1].key).replace(u' ', u'space')
270             # -----
271             self.hub.sendMessageEvent(text=u"Frame Ended (ID: {0}, Time:{1},
272                                     u"Selected key: {2}, Correct key:
273                                     {3})".format(
274                                     frame_index, timer.getTime(), key, frame[u'correct_keys_str']
275                                     ))
276             # -----
277             state = u'flip'
278
279         elif timer.getTime() >= frame[u'time']:                # is a timed frame
280             self.hub.sendMessageEvent(text=u"Frame Ended (ID: {0}, Time:{1})
281                                     ".format(
282                                     frame_index, timer.getTime()))
283             # -----
284             state = u'flip'
285
286         # -----
287         if kb.getKeys(keys=[u'escape', u'q', ]):
288             self.hub.sendMessageEvent(text=u"== EXPERIMENT ENDED BY USER ==
289
290
291             self.hub.quit()
292             window.close()
293             core.quit()
294             return 0
295
296         # -----
297         self.hub.clearEvents(u'all')
298
299     elif case(u'end'):
300         self.hub.sendMessageEvent(text=u"Ending Test (ID:{0} , Name: {1})".
301                                     format(
302                                     test_index, test[u'name']))
303         # -----
304         is_test_finish = True
305
306         # =====
307         tracker.setRecordingState(False)
308         # -----
309         test_count += 1
310
311         # =====
312         # Experiment exit

```

```

307         # =====
308         self.hub.sendMessageEvent(text=u"== EXPERIMENT ENDED NORMALLY == ")
309         window.close()
310         core.quit()
311
312
313     # =====
314     class Switch:
315         def __init__(self, value):
316             self._val = value
317
318         def __enter__(self):
319             return self
320
321         def __exit__(self, swt_type, value, traceback): # Allows traceback to occur
322             return False
323
324         def __call__(self, *mconds):
325             return self._val in mconds

```

## Script para crear la base de datos.

```
1  =====
2  -- SQLite Script generated manually
3  -- Date      : 29/01/2018
4  -- model     : saccadedb
5  -- author    : Christian Wiche
6  =====
7  pragma recursive_triggers=1;
8  pragma foreign_keys=1;
9
10
11 =====
12 -- MAIN CONFIG TABLE
13 =====
14 create table if not exists master (
15     mas_name          varchar(50)      not null,
16     mas_screen        int              not null    default 0,
17     mas_tracker       varchar(20)      not null    default 'None',
18     mas_monitor       varchar(50)      not null    default 'default',
19     mas_path          varchar(200)     not null    default './events',
20     primary key (mas_name)
21 );
22
23 =====
24 -- EXPERIMENT-RELATED TABLES
25 =====
26 -- Experiment: base table -----
27 create table if not exists experiment (
28     exp_code          varchar(10)      not null,
29     exp_name          varchar(50)      not null,
30     exp_version       varchar(10)      not null,
31     exp_description   text             null        default '',
32     exp_instructions  text             null        default '',
33     exp_comments      text             null        default '',
34     exp_date_creation timestamp        null        default current_timestamp,
35     exp_date_update   timestamp        null        default current_timestamp,
36     primary key (
37         exp_name,
38         exp_version
39     )
40 );
41 -- date update
42 create trigger if not exists trg_exp
43     after update of
44         exp_code,
45         exp_name,
46         exp_version,
47         exp_description,
48         exp_instructions,
49         exp_comments
50     on experiment
51     begin
52         update experiment
53             set exp_date_update = current_timestamp
54             where exp_code = new.exp_code;
55     end;
56 -- index
57 create unique index if not exists unq_exp on experiment (exp_code);
58
59 -----
60 -- Experiment: exp_dia -----
61 create table if not exists exp_dia (
62     exp_code          varchar(10)      not null,
63     dia_is_active     tinyint          not null    default 1,
64     dia_ask_age       tinyint          not null    default 1,
65     dia_ask_gender    tinyint          not null    default 1,
```

```

66     dia_ask_glasses      tinyint      not null      default 1,
67     dia_ask_eye_color    tinyint      not null      default 1,
68     primary key (exp_code)
69     constraint fk_exp_dia
70         foreign key (exp_code)
71         references experiment (exp_code)
72         on delete cascade
73         on update cascade
74 );
75 -- index
76 create index if not exists idx_exp_dia on exp_dia (exp_code asc);
77
78 -----
79 -- Experiment: exp_con -----
80 create table if not exists exp_con (
81     exp_code              varchar(10)    not null,
82     con_need_space        tinyint        not null      default 0,
83     con_is_random         tinyint        not null      default 0,
84     con_is_rest           tinyint        not null      default 0,
85     con_rest_period       int            not null      default 0,
86     con_rest_time         float          not null      default 0.0,
87     primary key (exp_code)
88     constraint fk_exp_con
89         foreign key (exp_code)
90         references experiment (exp_code)
91         on delete cascade
92         on update cascade
93 );
94 -- index
95 create index if not exists idx_exp_con on exp_con (exp_code asc);
96
97 =====
98 -- TEST-RELATED TABLES (EXPERIMENT CHILD)
99 =====
100 -- Test: base table -----
101 create table if not exists test (
102     exp_code              varchar(10)    not null,
103     tes_index             int            not null      default 1,
104     tes_name              varchar(50)    not null      default 'Unnamed',
105     tes_description        text          null          default '',
106     tes_quantity          int            not null      default 1,
107     primary key (
108         exp_code,
109         tes_index
110     )
111     constraint fk_tes
112         foreign key (exp_code)
113         references experiment (exp_code)
114         on delete cascade
115         on update cascade
116 );
117 -- index
118 create index if not exists idx_tes on test (exp_code asc, tes_index asc);
119
120 -----
121 -- Test: tes_frame -----
122 create table if not exists frame (
123     exp_code              varchar(10)    not null,
124     tes_index             int            not null      default 1,
125     fra_index             int            not null      default 1,
126     fra_name              varchar(50)    not null      default 'Unnamed',
127     fra_color             varchar(20)    not null      default 'black',
128     fra_is_task           tinyint        not null      default 0,
129     fra_keys_allowed      text,
130     fra_keys_selected     text,
131     fra_time              float          not null      default 0.0,
132     primary key (
133         exp_code,

```

```

134         tes_index,
135         fra_index
136     )
137     constraint fk_fra
138         foreign key (exp_code, tes_index)
139             references test (exp_code, tes_index)
140             on delete cascade
141             on update cascade
142 );
143 -- index
144 create index if not exists idx_fra on frame (exp_code asc, tes_index asc, fra_index asc);
145
146 -- Test: tes_fra_object -----
147 create table if not exists component (
148     exp_code          varchar(10)      not null,
149     tes_index         int              not null    default 1,
150     fra_index         int              not null    default 1,
151     com_index         int              not null    default 1,
152     com_name          varchar(50)      not null    default 'Unnamed',
153     com_units         varchar(10)      not null    default 'deg',
154     com_pos_x         float            not null    default 0.0,
155     com_pos_y         float            not null    default 0.0,
156     com_orientation   float            not null    default 0.0,
157     com_size          float            not null    default 1.0,
158     com_shape         varchar(20)      not null    default 'square',
159     com_image         blob              null,
160     com_color         varchar(20)      not null    default 'white',
161     primary key (
162         exp_code,
163         tes_index,
164         fra_index,
165         com_index
166     )
167     constraint fk_obj
168         foreign key (exp_code, tes_index, fra_index)
169             references frame (exp_code, tes_index, fra_index)
170             on delete cascade
171             on update cascade
172 );
173 -- index
174 create index if not exists idx_obj on component (exp_code asc, tes_index asc, fra_index asc,
        com_index asc);

```