PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ

FACULTAD DE CIENCIAS E INGENIERÍA

Desarrollo de una herramienta informática en MATLAB para la estandarización de la toma de datos en estudios de vida pública.

Tesis para optar por el Título de Ingeniero Civil, que presenta el bachiller:

José Antonio Mallma León

ASESOR: Félix Israel Cabrera Vega

Lima, diciembre de 2016

ÍNDICE

ANEXO A

Guía de usuario

Plife

La presente guía de usuario tiene como objetivo detallar el correcto uso del programa PLife. Este programa ha sido desarrollado como parte del proyecto de tesis del bachiller con mención en ingeniería civil José Antonio Mallma León. El programa ha sido desarrollado en la plataforma MATLAB v2015a.

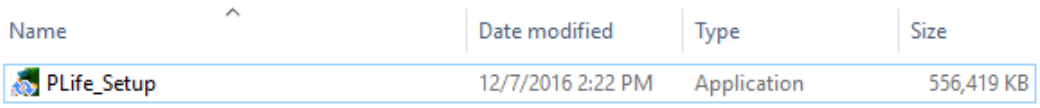
El programa PLife ha sid desarrollado con el objetivo de proponer una herramienta que permita estandarizar los datos registrados en estudios de vida pública. El programa se basa en las herramientas identificadas por Jan Gehl en su libro How to study Public Life.

La presente guía se divide en ocho capítulos. El primer capítulo cubre los pasos necesarios para la instalación del programa. El segundo capítulo se enfoca en la interfaz principal. Los seis restantes capítulos contemplan las seis herramientas implementadas en el programa: Mapping, Tracing, Tracking, Counting, Speed Test, Diary.

Revisión – Diciembre 2016

Capítulo 1 – Instalación

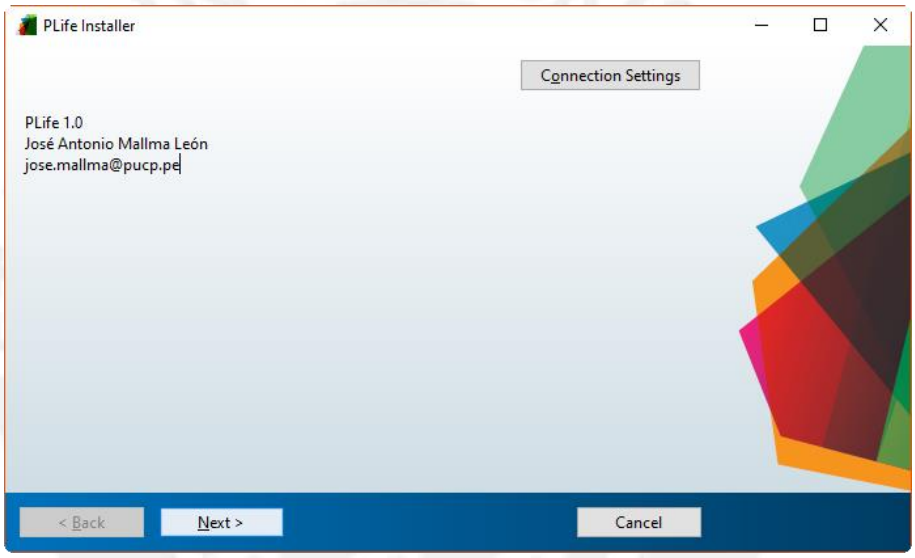
Junto con este documento usted debió recibir una copia del archivo de instalación del programa PLife.



Instalador PLife

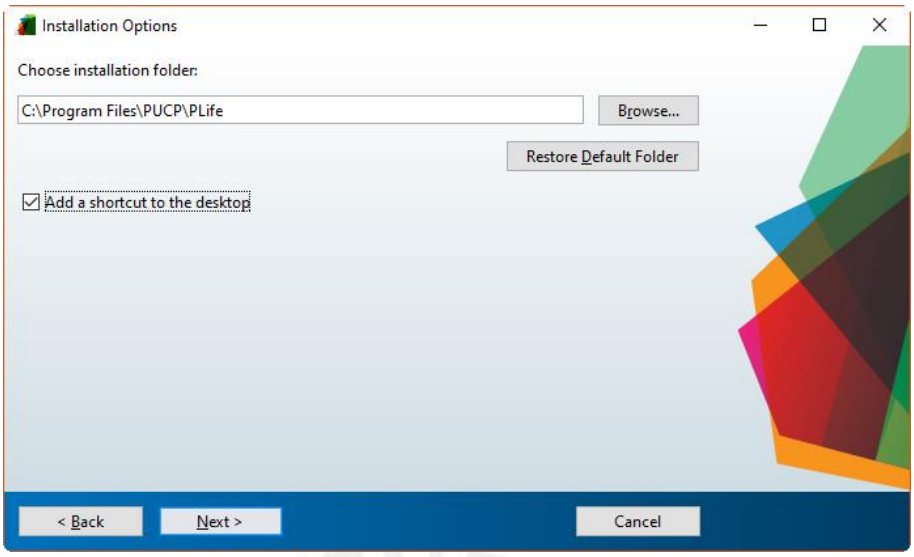
Ejecute el instalador. Dependiendo de la configuración de su equipo es posible que se solicite autorización para instalar el programa.

Al iniciar se mostrará la siguiente ventana.



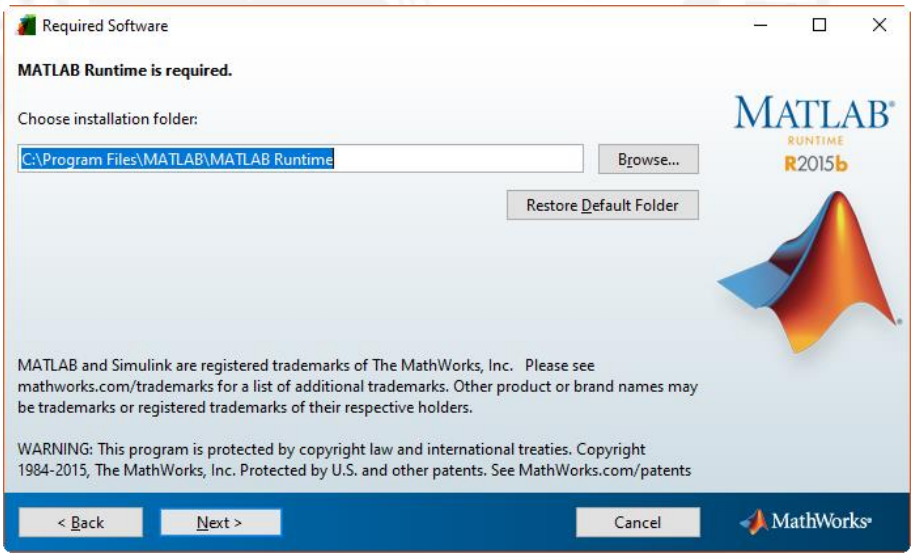
Instalador PLife

Presione el botón siguiente y se solicitará la ruta para instalar el programa. Se recomienda seleccionar una carpeta de fácil acceso ya que los resultados del programa serán guardados en dicha carpeta.



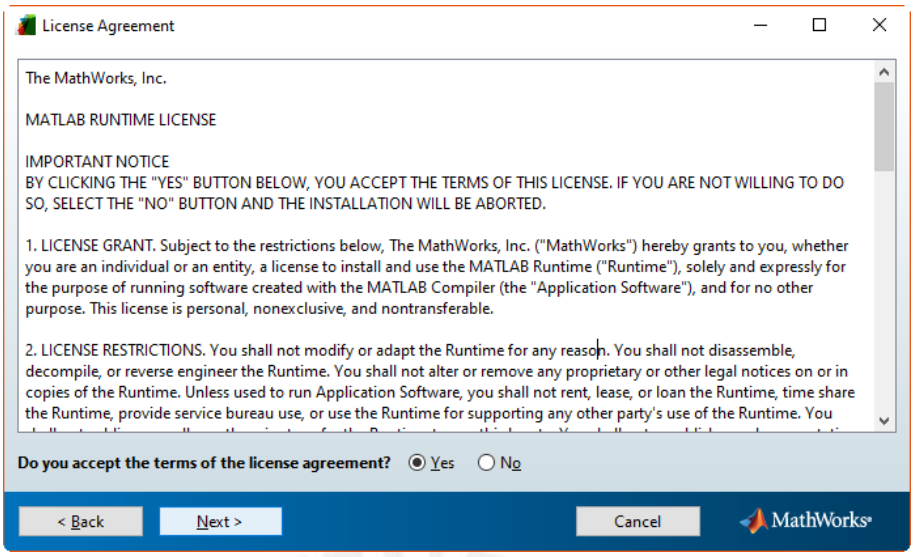
Instalador PLife

Al hacer clic en siguiente se solicitará especifique la ruta para la instalación de MATLAB Runtime. Este es un programa aparte distribuido oficialmente por MATLAB. Este programa permite ejecutar aplicación sin necesidad de contar con alguna versión de MATLAB. Además, este compilador garantiza que todas las funciones implementadas se ejecuten correctamente. Se sugiere no modificar la ruta de instalación, pero queda a decisión del usuario.



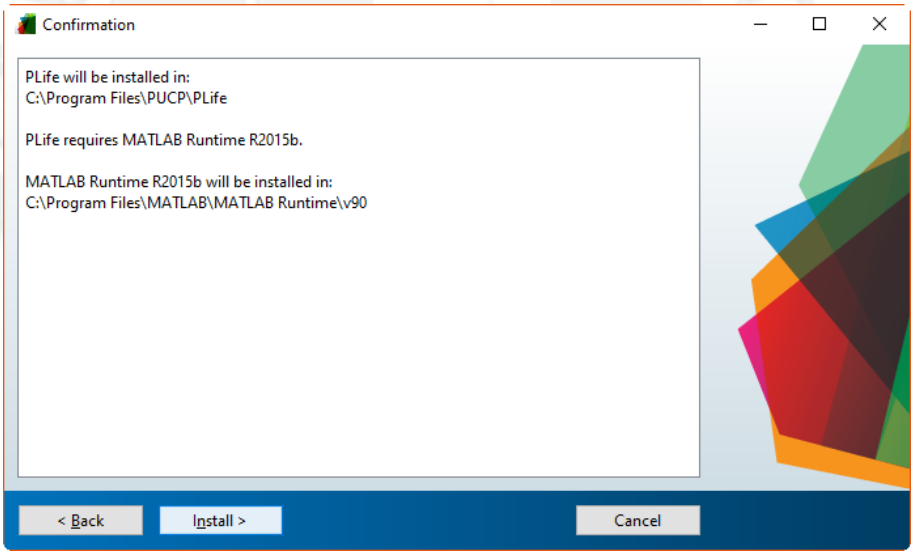
Instalador MATLAB Runtime

Luego se solicitará que acepte los términos y condiciones para la instalación de MATLAB Runtime. Puede revisar dicha información desde el instalador.



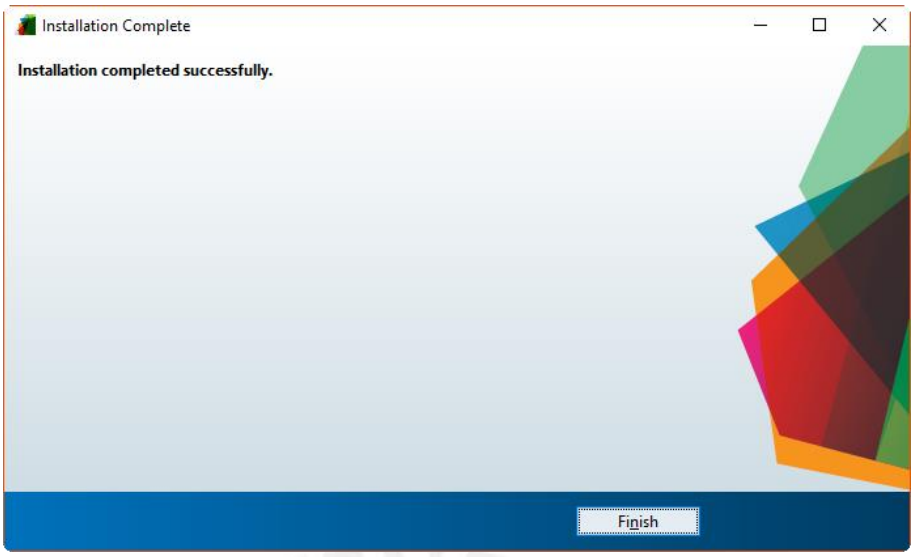
Términos y condiciones de MATLAB Runtime

Finalmente se mostrará las carpetas seleccionadas para dar inicio al proceso de instalación



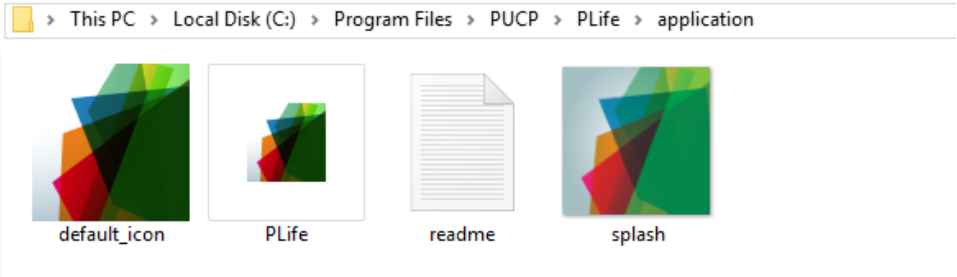
Instalador PLife

La instalación puede tomar varios minutos. Será informado una vez se concluya con esta.



Instalación completada

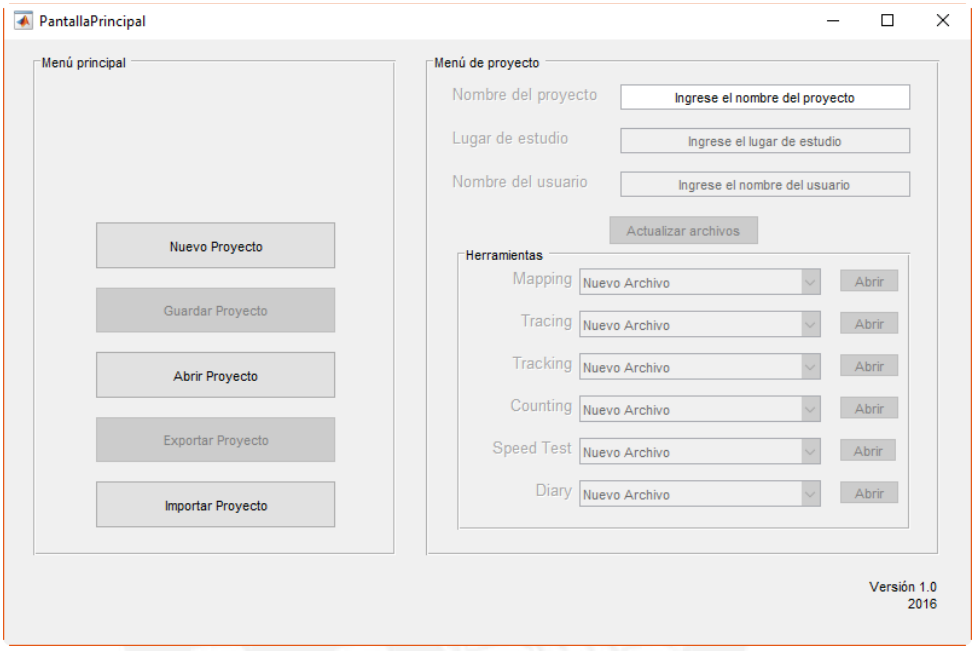
Ahora el programa está listo para usarse. Los archivos generados por el programa serán guardados dentro de la carpeta “application”.



Carpeta de instalación

Capítulo 2 – Pasos iniciales

Para iniciar el programa abra el acceso directo que se ha creado en su escritorio, o desde la carpeta donde instaló el programa. Luego de ejecutar el programa se muestra la siguiente pantalla. La pantalla principal se divide en dos áreas, en la parte izquierda se encuentra el panel del Menú Principal y en la parte derecha el panel del Menú de Proyecto.



Pantalla inicial

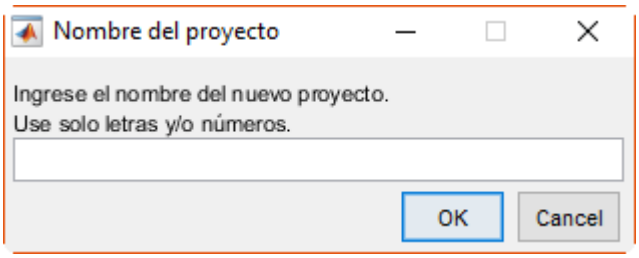
Dentro del Menú Principal se encuentran cinco botones, de los cuales, al iniciar la aplicación, se encuentran habilitados tres. A continuación, se detalla los pasos a seguir para iniciar con el programa.

1. Crear un nuevo proyecto.

   PLife trabaja en base a proyectos en los cuales se almacena la información asociada a un estudio en particular. De esta manera se facilita la organización de los datos ingresados.

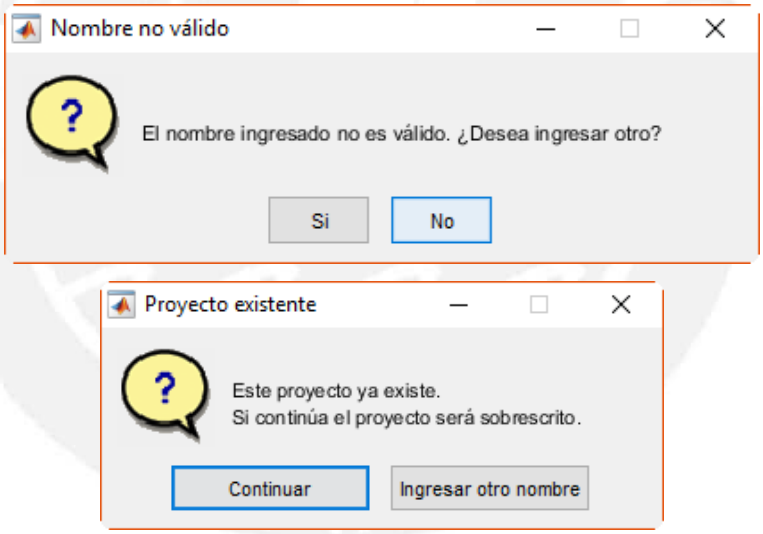
   Los proyectos son guardados en la carpeta “Proyectos” que se crea automáticamente en la carpeta donde ha sido instalado el programa. Cada proyecto cuenta con una carpeta propia en la cual se almacena los datos de dicho proyecto.

Al presionar el botón “Nuevo Proyecto” se solicita el nombre que será asignado al proyecto. El nombre debe contener únicamente caracteres alfanuméricos, espacios en blanco o guiones.



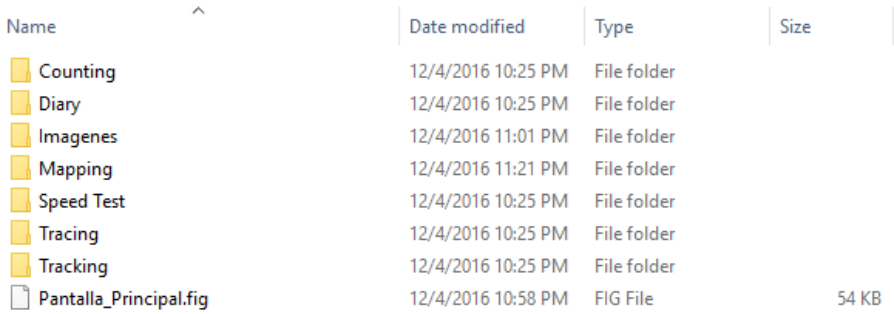
Ingresar nombre del proyecto

En caso el nombre ingresado no sea válido o ya exista un proyecto con el mismo nombre, el programa informará al respecto. En el primer caso se dará la opción de volver a ingresar otro nombre. En el segundo caso se pregunta al usuario si desea ingresar otro nombre o sobrescribir el proyecto existente. En caso el usuario decida sobrescribir el proyecto, **todos los datos existentes serán eliminados**.



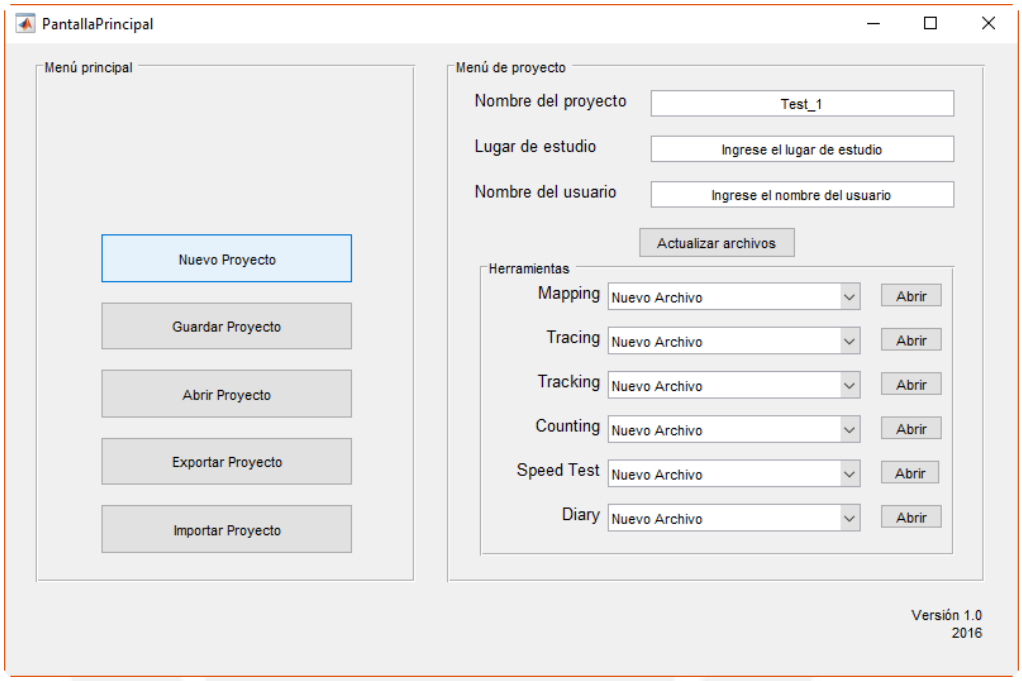
Notificaciones al crear un proyecto

Luego de ingresar un nombre válido se creará la carpeta del proyecto y dentro de ella siete carpetas: una carpeta “Imágenes” en donde se almacenará todas las imágenes que el usuario ingrese en el proyecto, y seis carpetas asociadas a las herramientas habilitadas en el programa. Además de las carpetas también se crea un archivo de nombre Pantalla_Principal.fig, este archivo es utilizado para cargar el proyecto y no debe ser manipulado por el usuario.



Carpeta del proyecto.

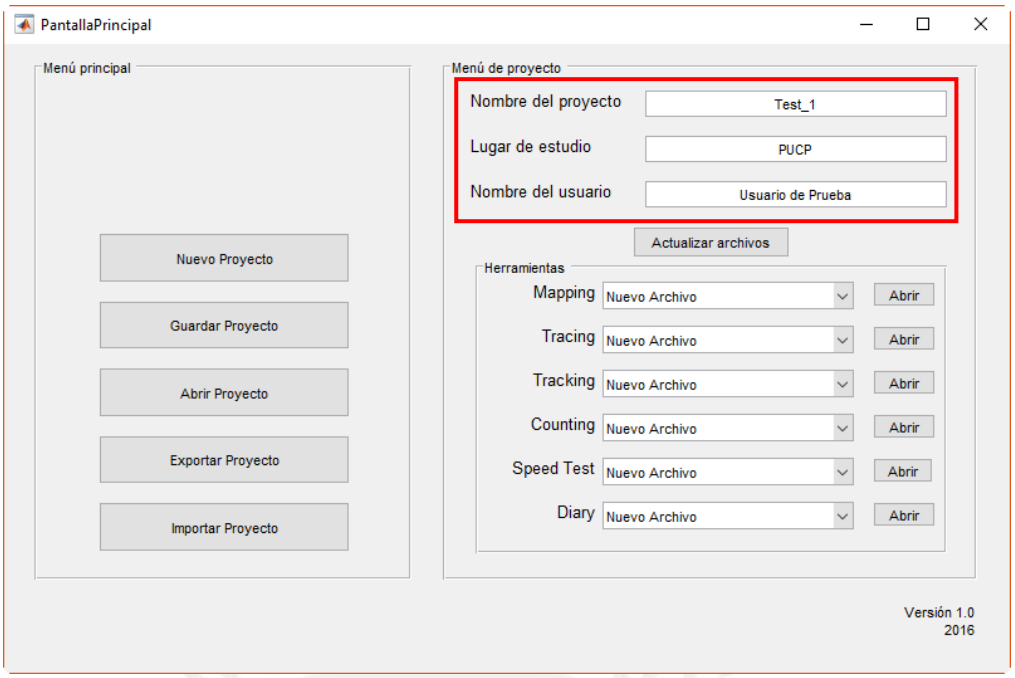
Al mismo tiempo se habilitarán todos los elementos de la pantalla principal.



Pantalla principal habilitada.

2. Ingresar información general del proyecto

En el panel derecho de la pantalla se habilitarán tres campos de texto. El primer campo “Nombre del Proyecto” muestra el nombre ingresado por el usuario y no puede ser modificado. Los dos campos restantes: Lugar de estudio y Nombre del usuario, pueden ser editados libremente y de manera opcional. El contenido ingresado será mostrado en todos archivos creados dentro del proyecto.

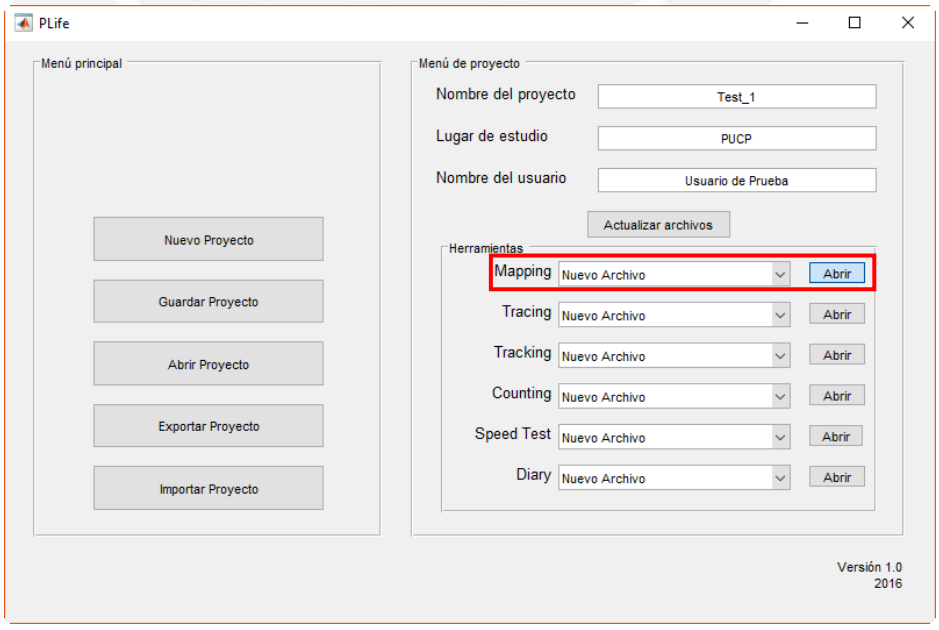


Información del proyecto.

3. Crear un nuevo archivo y abrir uno existente

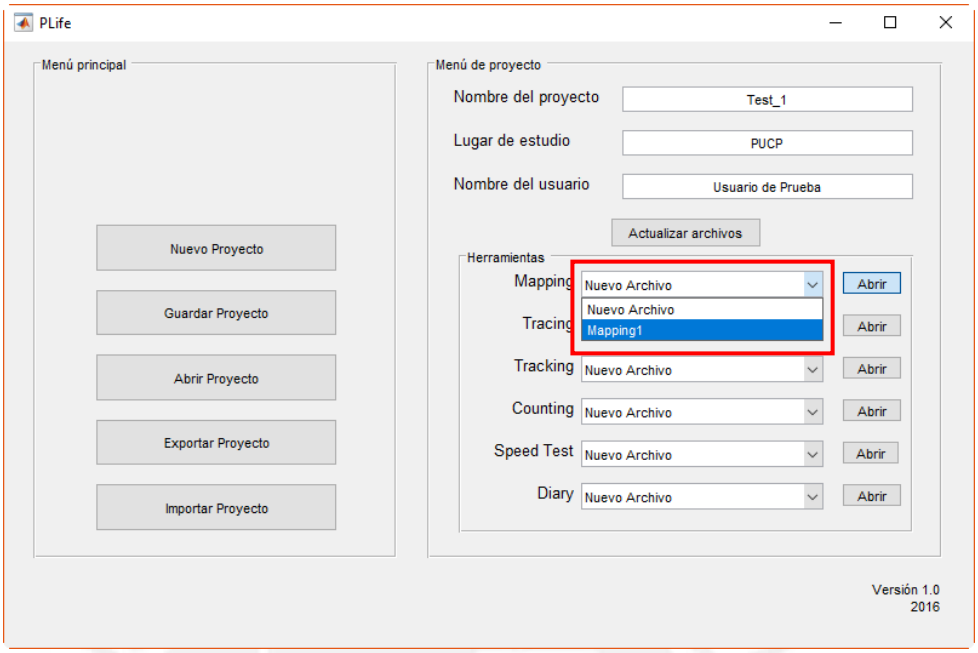
En el panel del proyecto se encuentra un subpanel de herramientas. En este se cuenta con seis menús desplegables y seis botones, uno para cada herramienta implementada.

Para crear un nuevo archivo se debe tener seleccionado el ítem Nuevo archivo en el menú desplegable y luego presionar el botón “Abrir”.



Crear nuevo archivo.

Si se desea abrir un nuevo archivo, este debe aparecer en el menú desplegable respectivo. Bastará con seleccionarlo y luego pulsar el botón Abrir.



Crear nuevo archivo.

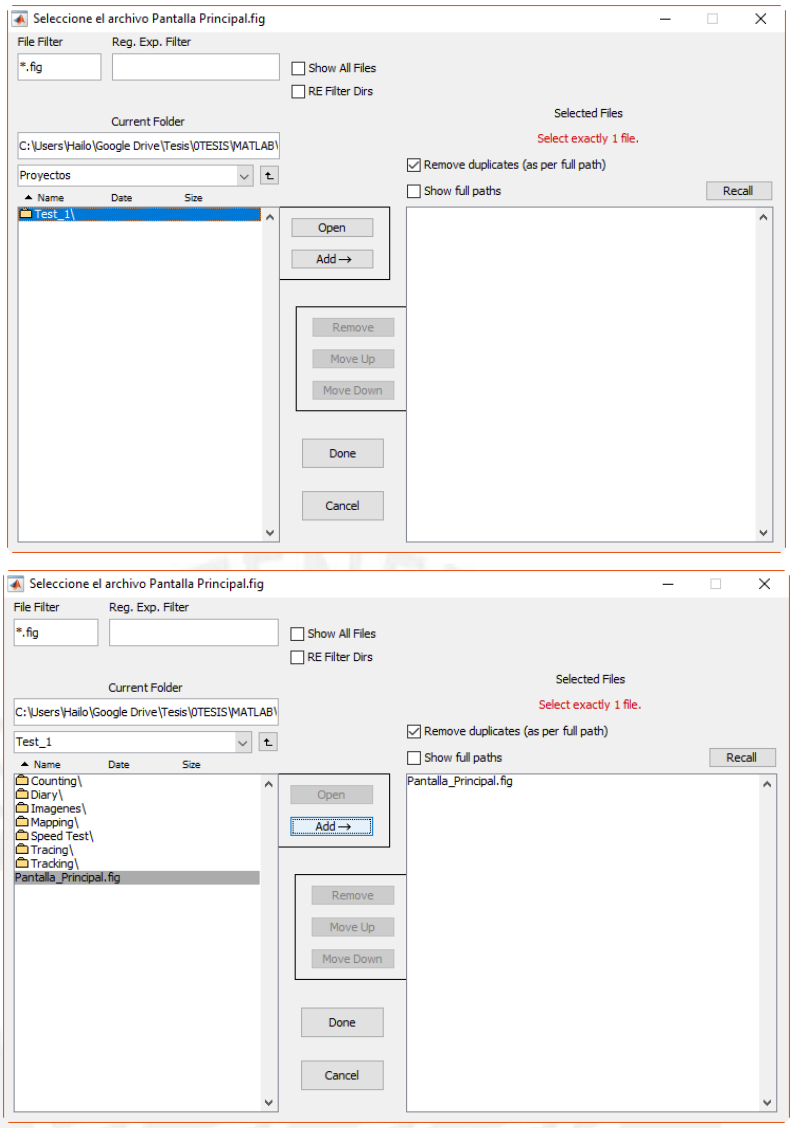
Luego de crear un nuevo archivo o abrir uno existente la pantalla principal se cerrará y se abrirá la interfaz de la herramienta respectiva.

4. Guardar un proyecto

Para poder continuar trabajando en un proyecto luego de cerrar el programa es necesario guardarlo. Para ello se cuenta con el botón “Guardar Proyecto” que automáticamente guardará la información el proyecto.

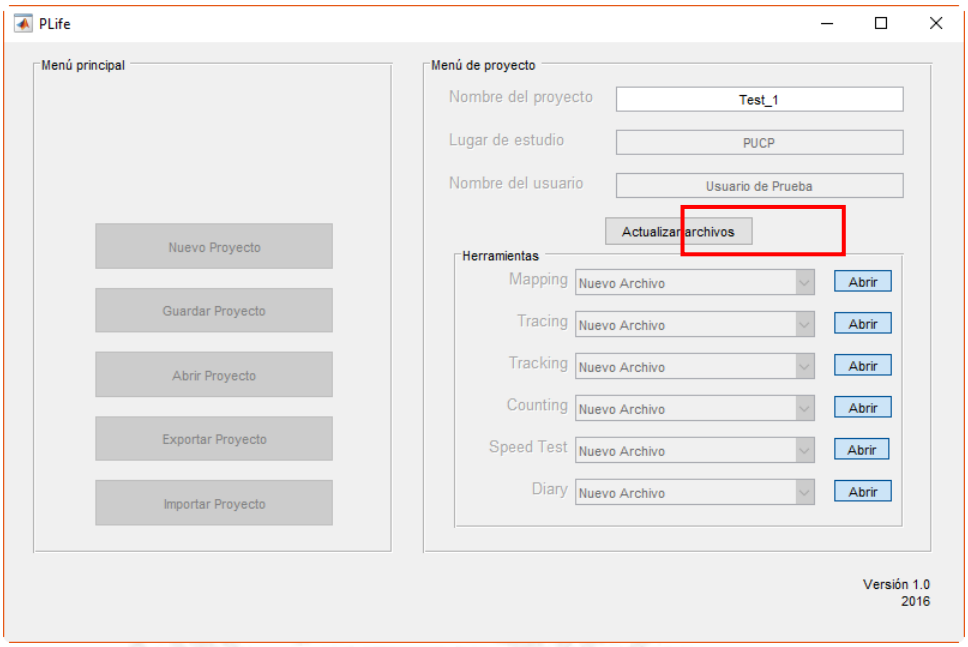
5. Abrir un proyecto

Para cargar un proyecto existente se debe presionar el botón Abrir proyecto al iniciar la aplicación. Esto habilitará una ventana en donde se muestran las carpetas existentes dentro de la carpeta “Proyectos”. Para cargar el proyecto deseado es necesario seleccionar el archivo Pantalla_Principal.fig que se encuentra dentro de la carpeta del respectivo proyecto y presionar el botón Done.



Seleccionar archivo Pantalla_Principal para abrir un proyecto

De esta manera se cargará la pantalla principal del proyecto. Es importante indicar que la pantalla se cargará con todos los elementos deshabilitados, excepto el botón Actualizar Archivos que se encuentra en el panel derecho. Es necesario presionar dicho botón para cargar la información restante.



Actualizar archivos para finalizar carga de proyecto.

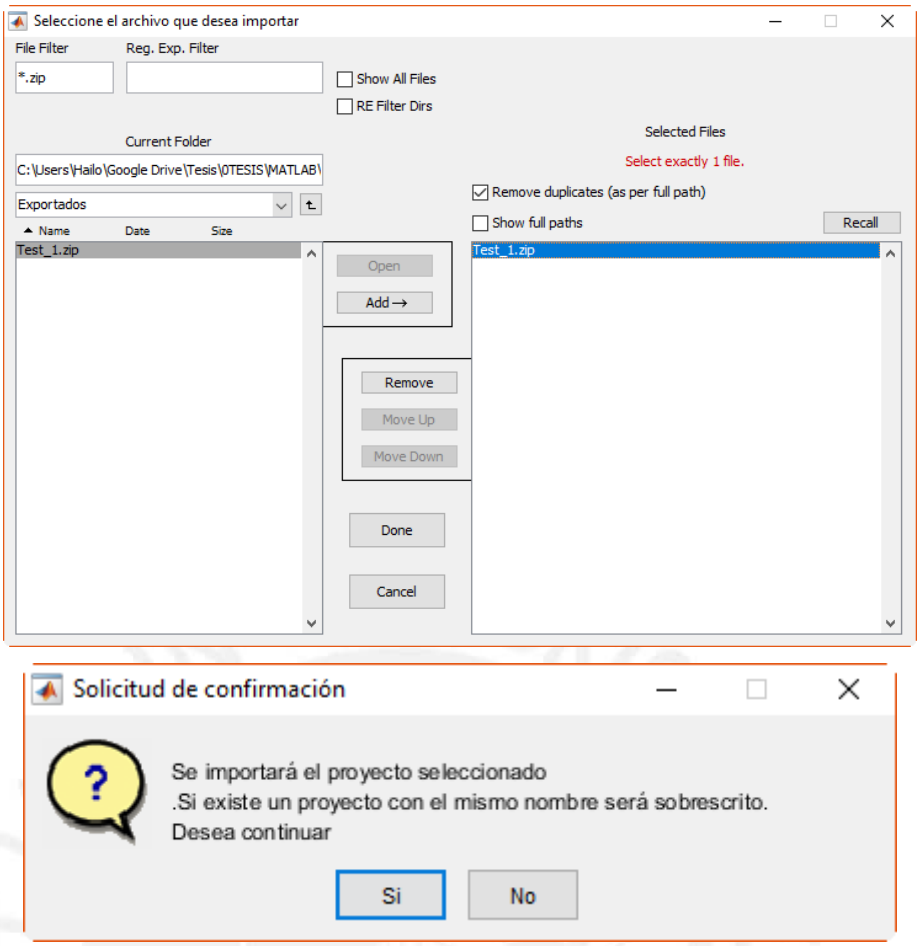
6. Exportar proyecto

   Con el objetivo de poder compartir los proyectos creados en una computadora para que estos puedan ser usados en otro equipo se cuenta con el botón Exportar Proyecto. Al presionar dicho botón se creará un archivo ZIP en la carpeta "Exportados" dentro de la carpeta donde fue instalado el programa.

7. Importar proyecto

   En la pantalla principal se cuenta con un botón para importar un proyecto que ha sido exportado. Al presionar el botón el programa solicitara que se seleccione el archivo en formato ZIP que fue generado al exportar el proyecto por el usuario emisor.

   IMPORTANTE: Si en el equipo receptor existe un proyecto del mismo nombre, el programa consultará si el usuario desea importar proyecto, en tal caso el proyecto existente será sobrescrito y se perderá la información.

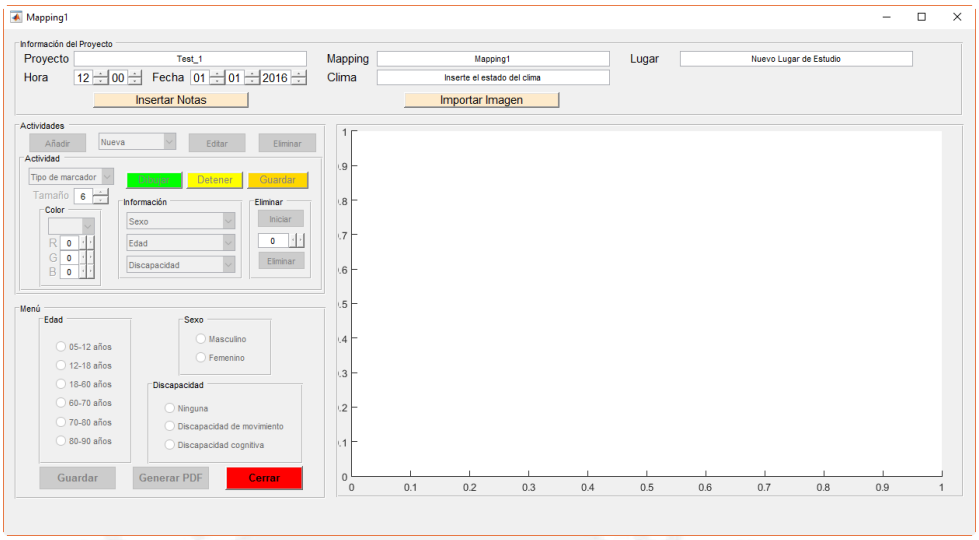


Importar proyecto.

Luego de importar el proyecto se mostrará un mensaje de información. Para acceder al proyecto este debe ser abierto desde el botón Abrir Proyecto.

Capítulo 3 – Mapping

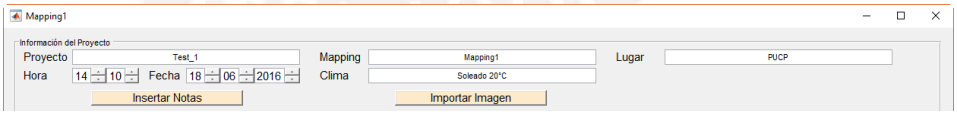
Al crear un nuevo archivo de la herramienta Mapping se abre la siguiente pantalla.



Pantalla inicial Mapping.

1. Editar información del proyecto

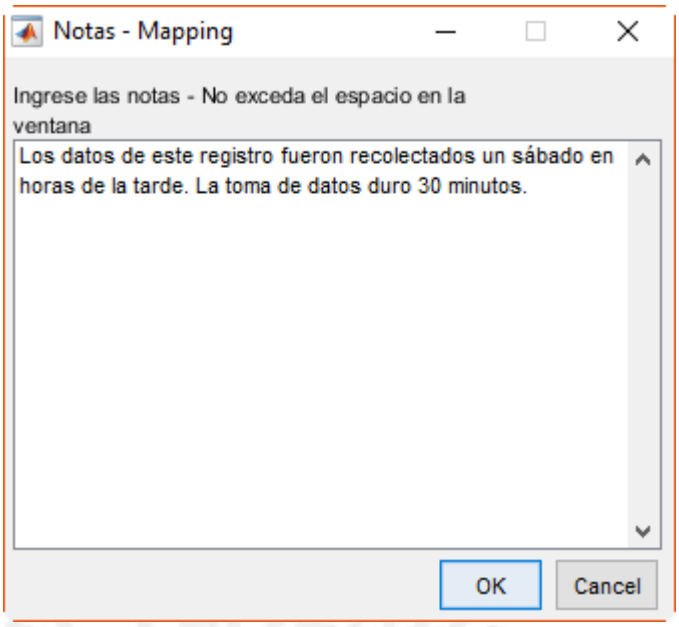
   En la parte superior se encuentra el panel Información del Proyecto. Dentro del panel hay cuatro campos de texto. El primero muestra el nombre del proyecto; el segundo, el nombre del archivo; y el tercero, el nombre del lugar analizado. Estos recuadros no son editables y muestran la información ingresada en la pantalla principal. El cuarto recuadro permite ingresar las condiciones climáticas registradas. En el mismo panel se cuenta con botones para definir la hora y fecha del registro. Toda la información ingresada se mostrará en el reporte generado por el programa.



Información del Proyecto.

2. Ingresar notas de texto.

   También se provee un espacio para ingresar notas de texto. Para ello se debe pulsar el botón Insertar Notas y se habilitará una ventana para ingresar el texto. El usuario debe tener cuidado de no exceder el espacio disponible en pantalla, caso contrario el texto no se mostrará apropiadamente en el reporte.

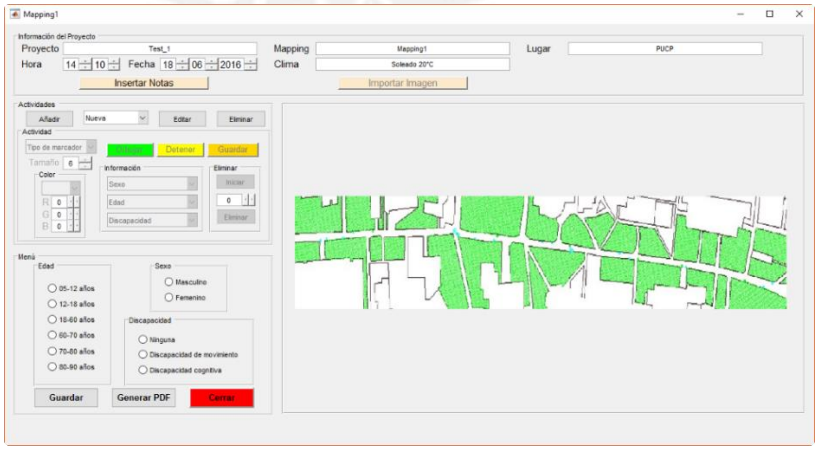


Notas de texto.

3. Importar croquis

Esta herramienta necesita que un croquis del espacio analizado sea importado. Para ello, el usuario debe contar con dicho croquis en un archivo en formato JPG. Para importar la imagen se debe presionar el botón “Importar Imagen”. Luego se habilita una ventana para seleccionar el archivo deseado. Luego de seleccionar la imagen esta se muestra en la pantalla de la aplicación.

IMPORTANTE: Por defecto se muestra la carpeta Imágenes dentro del proyecto. Si el usuario selecciona un archivo que no esté dentro de esta carpeta el programa creará una copia de la imagen en la carpeta “Imágenes”. De esta manera, todas las imágenes del proyecto son almacenadas en la misma carpeta para su posterior uso desde otras aplicaciones.

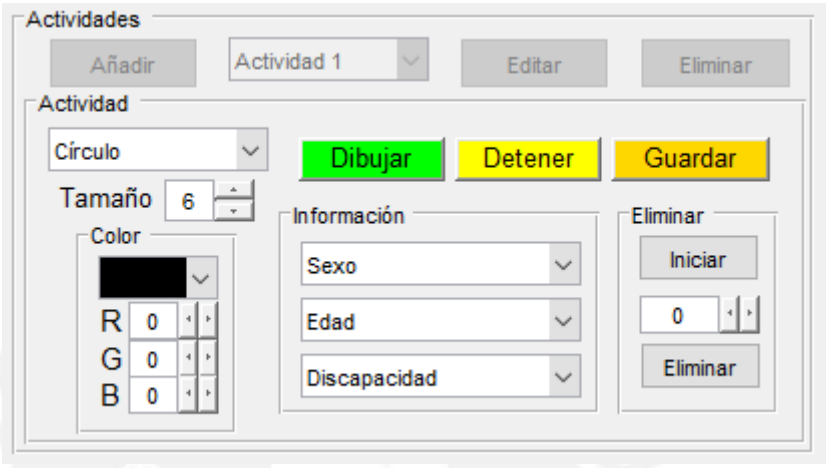


Importar croquis.

4. Actividades

4.1. Crear actividad

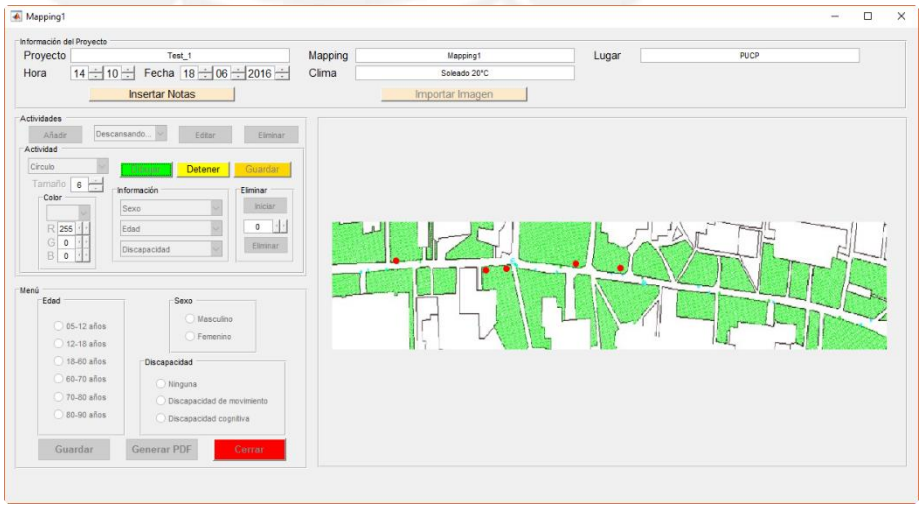
Para crear una actividad pulse el botón “Añadir”. El programa solicitará que ingrese el nombre de la nueva actividad. Solo se permite caracteres alfanuméricos en el nombre. Luego de crear la actividad se habilitará el panel “Actividad”.



Panel Actividades.

4.2. Dibujar actividad

Las actividades son graficadas como puntos sobre el croquis. Para ubicar los puntos presione el botón Dibujar. Todos los elementos de la pantalla se deshabilitarán excepto el botón Detener. Mientras no presione el botón Detener podrá dibujar puntos sobre el croquis haciendo clic sobre la imagen.



Dibujar puntos.

Al pulsar el botón Detener se volverá a habilitar los elementos de la pantalla y ya no podrá dibujar puntos sobre el croquis. Puede volver a presionar el botón Dibujar si lo requiere.

4.3. Personalizar

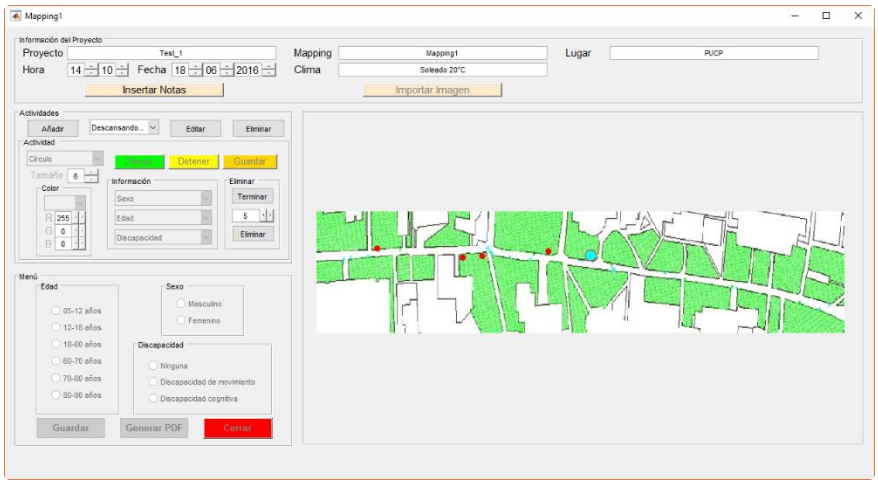
A los puntos de cada actividad se les asigna un formato que consiste del tipo de marcador, el tamaño del marcador y el color del marcador. Estas propiedades pueden ser modificadas desde el panel Actividad. Seleccione el marcador deseado del menú desplegable. El tamaño del marcador puede ser variado usando los botones bajo el menú desplegable. Con respecto a los colores se provee una serie de colores predefinidos que pueden ser seleccionados desde el menú desplegable en el subpanel Color. Sin embargo, estos no son los únicos colores disponibles ya que puede generar cualquier combinación variando las componentes RGB.

Personalizar puntos.

4.4. Eliminar puntos

En el mismo panel, se cuenta con el subpanel Eliminar. Al presionar el botón iniciar se habilitará la herramienta para borrar puntos de la actividad seleccionada. Usando los botones del subpanel puede desplazarse entre los puntos dibujados. El punto activo se resaltará sobre el croquis. Al pulsar el botón eliminar se borrará el punto activo. Para salir de la opción Eliminar Puntos pulse el botón Terminar.

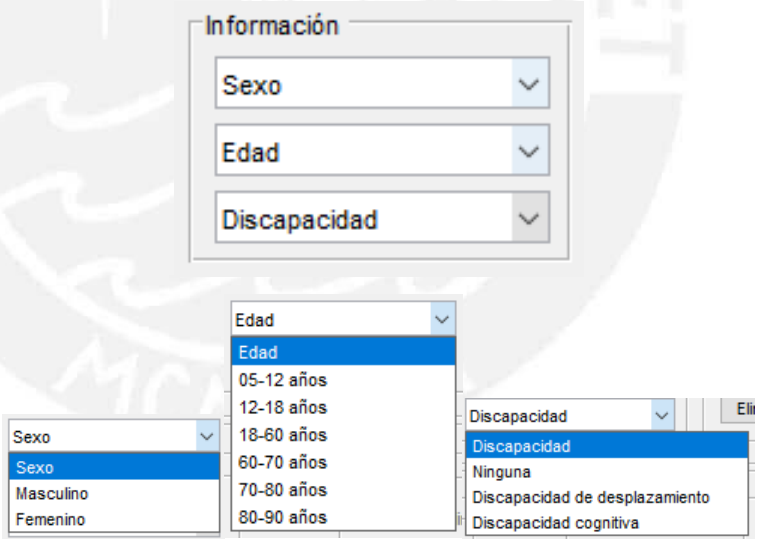
IMPORTANTE: Los puntos eliminados no podrán ser recuperados.



Eliminar puntos.

4.5. Información adicional

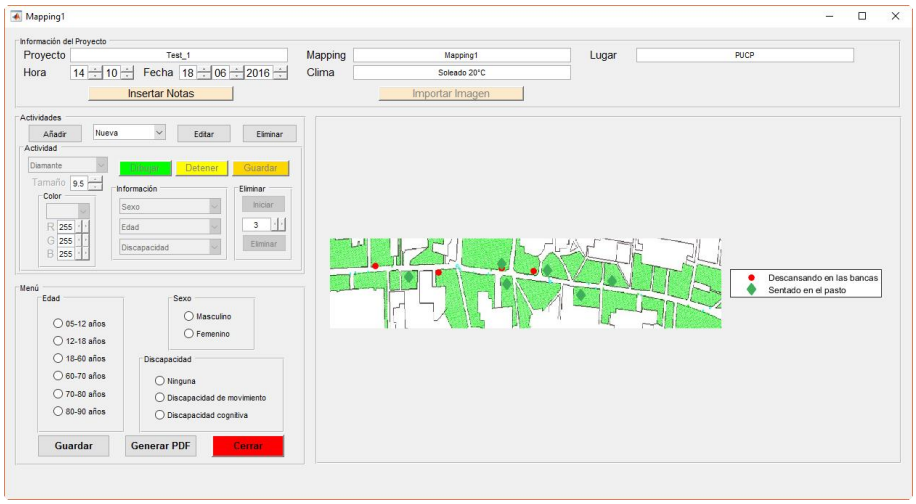
También puede asignar información adicional a los puntos ingresados. Los criterios asignables son sexo, rango de edad o la presencia de discapacidades de las personas registradas. Especificar esta información permite que el programa pueda elaborar reportes personalizados. Esta opción se verá más adelante.



Información adicional.

4.6. Guardar actividad

Luego de ingresar toda la información de la actividad presione el botón Guardar. El programa guardará la información ingresada y mostrará en el croquis todas las actividades existentes. El panel de Actividad se deshabilitará y no podrá editar ninguna actividad.



Guardar actividad.

4.7. Editar actividad

Si se desea editar una actividad existente debe seleccionarla en el menú desplegable en el panel de Actividades y luego presionar el botón Editar. Se habilitará el panel Actividad con la información de la actividad seleccionada.

4.8. Eliminar actividad

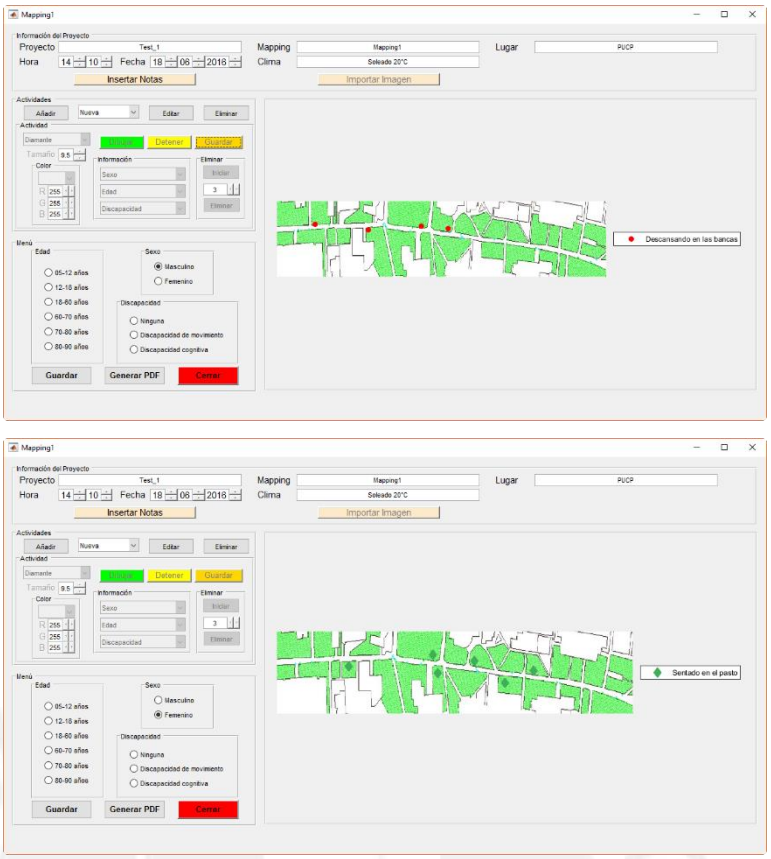
Para eliminar una actividad, selecciónela en el menú desplegable y presione el botón eliminar. Se solicitará la confirmación del usuario.

IMPORTANTE: Las actividades eliminadas no podrán ser recuperadas.

5. Visualización

En el panel inferior se cuenta con varios botones que permiten filtrar la información que se muestra en pantalla. Se mostrará en pantalla aquellas actividades que cumplan con los criterios de sexo, edad y presencia de discapacidades definidos.

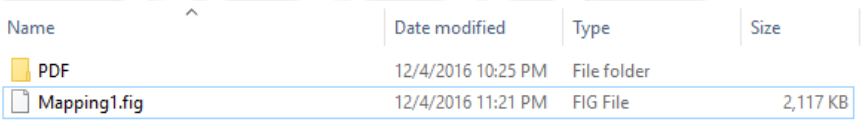
IMPORTANTE: En cada criterio, los estados de no marcar ningún botón o marcar todos son equivalentes.



Criterios de visualización.

6. Guardar archivo

Para guardar el archivo pulse el botón Guardar. El programa creará un archivo dentro de la carpeta asignada a la herramienta en la carpeta del proyecto. Este archivo no debe ser manipulado por el usuario.

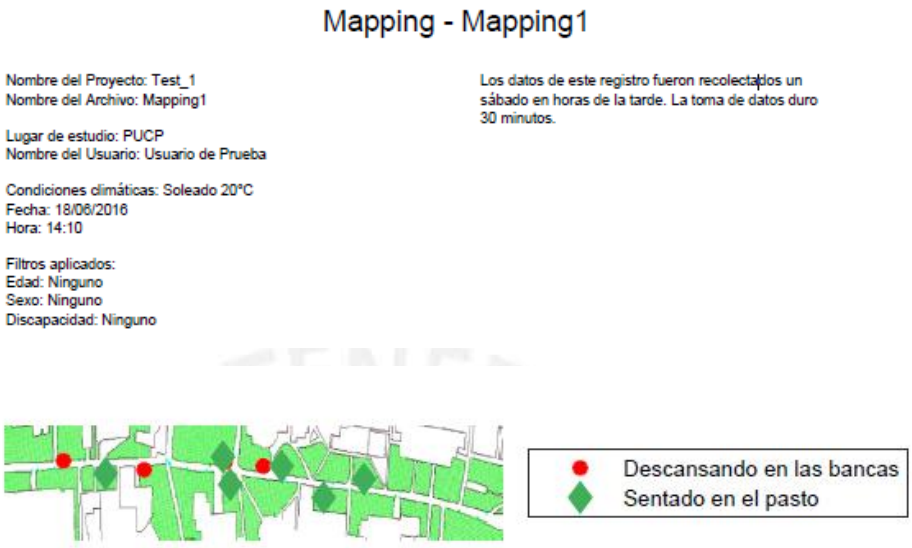


Guardar archivo.

7. Generar reportes

El programa permite generar reportes en formato PDF. Para ello basta con presionar el botón “Generar PDF”. El programa automáticamente generará un archivo en formato PDF dentro de la carpeta PDF que se muestra en la imagen anterior. La información mostrada en el reporte coincide con los criterios de visualización activos.

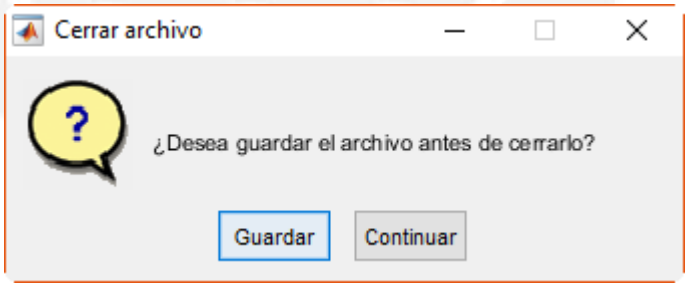
El archivo se abrirá automáticamente luego de ser creado. El programa está diseñado para no sobrescribir los reportes generados, de esta manera se pueden generar reportes con distintas configuraciones de visualización.



Reporte PDF.

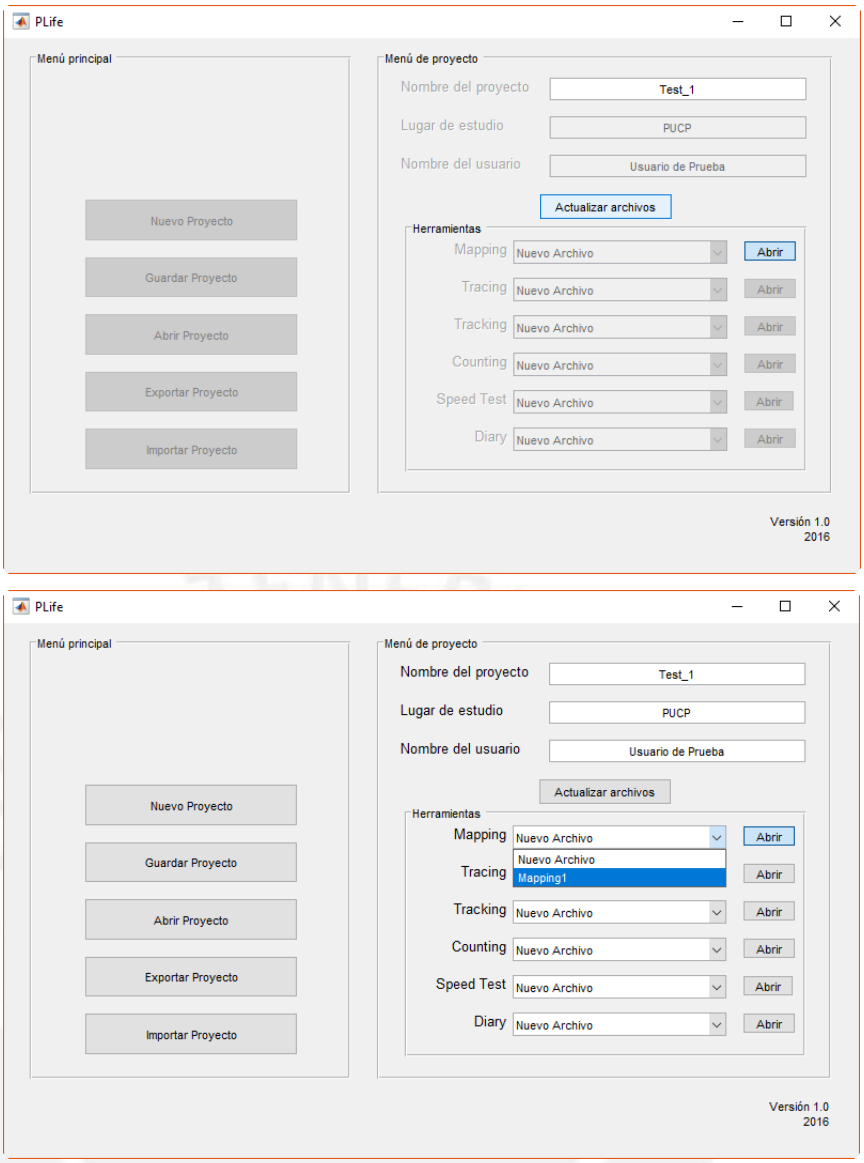
8. Cerrar herramienta

Finalmente, para regresar a la pantalla principal del programa presione el botón Cerrar. El programa consultará si desea guardar el archivo o si desea continuar sin guardar. Luego se cerrará la ventana de la herramienta y se abrirá la pantalla principal.



Cerrar herramienta.

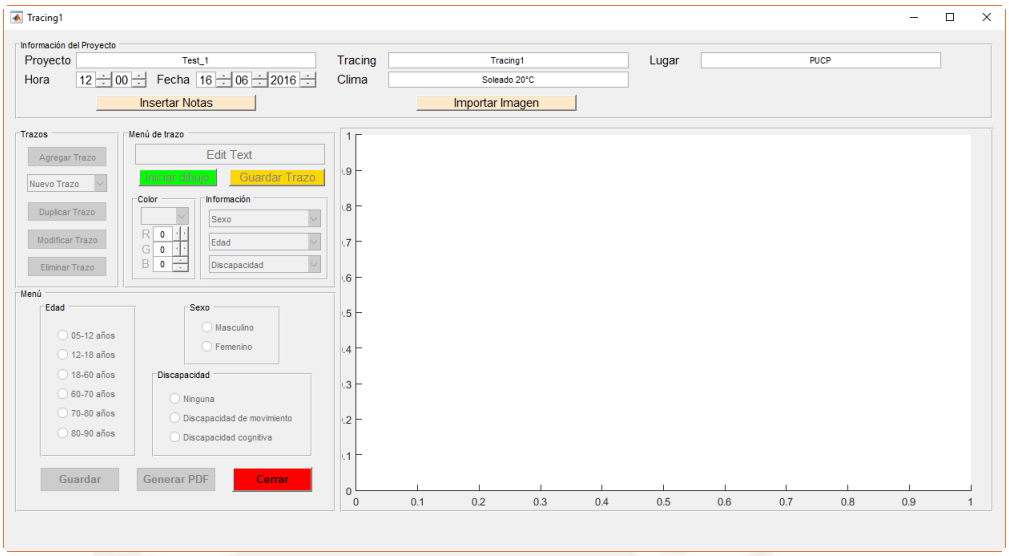
La ventana principal se abrirá con todos los elementos deshabilitados. Presione el botón Actualizar Archivos para habilitar la ventana. Si el archivo fue guardado, ahora este será visible en su respectivo menú desplegable.



Habilitar Pantalla Principal.

Capítulo 4 – Tracing

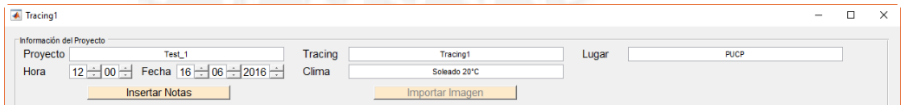
Al crear un nuevo archivo de la herramienta Tracing se abre la siguiente pantalla.



Pantalla inicial Tracing.

1. Editar información del proyecto

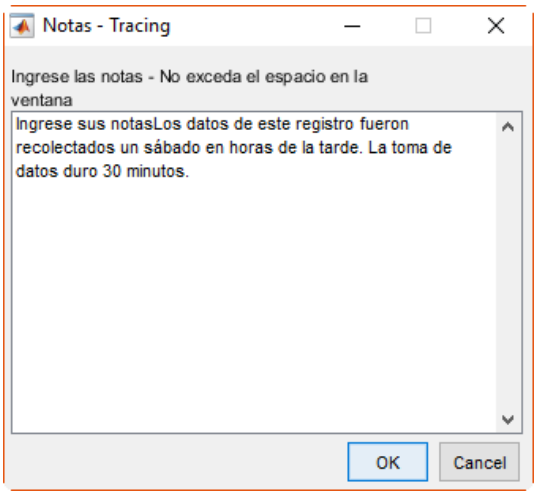
   En la parte superior se encuentra el panel Información del Proyecto. Dentro del panel hay cuatro campos de texto. El primero muestra el nombre del proyecto; el segundo, el nombre del archivo; y el tercero, el nombre del lugar analizado. Estos recuadros no son editables y muestran la información ingresada en la pantalla principal. El cuarto recuadro permite ingresar las condiciones climáticas registradas. En el mismo panel se cuenta con botones para definir la hora y fecha del registro. Toda la información ingresada se mostrará en el reporte generado por el programa.



Información del Proyecto.

2. Ingresar notas de texto.

   También se provee un espacio para ingresar notas de texto. Para ello se debe pulsar el botón Insertar Notas y se habilitará una ventana para ingresar el texto. El usuario debe tener cuidado de no exceder el espacio disponible en pantalla, caso contrario el texto no se mostrará apropiadamente en el reporte.

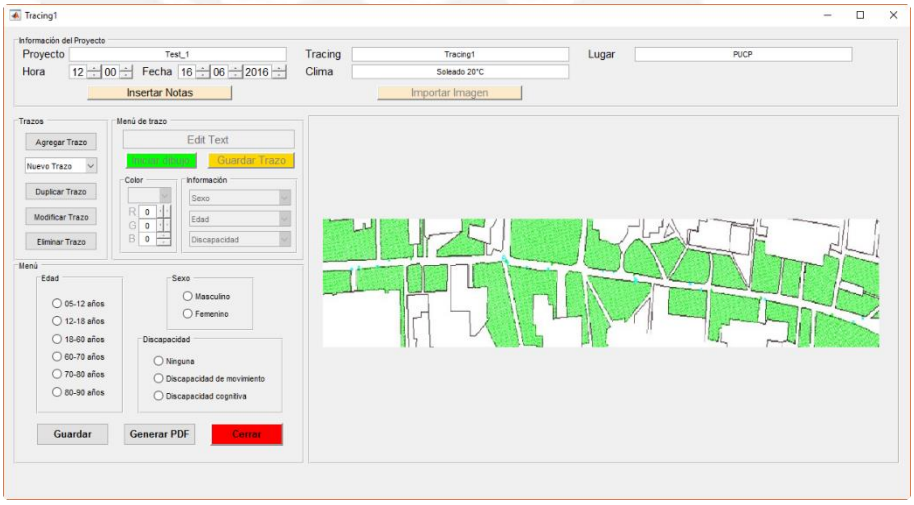


Notas de texto.

3. Importar croquis

Esta herramienta necesita que un croquis del espacio analizado sea importado. Para ello, el usuario debe contar con dicho croquis en un archivo en formato JPG. Para importar la imagen se debe presionar el botón “Importar Imagen”. Luego se habilita una ventana para seleccionar el archivo deseado. Luego de seleccionar la imagen esta se muestra en la pantalla de la aplicación.

IMPORTANTE: Por defecto se muestra la carpeta Imágenes dentro del proyecto. Si el usuario selecciona un archivo que no esté dentro de esta carpeta el programa creará una copia de la imagen en la carpeta “Imágenes”. De esta manera, todas las imágenes del proyecto son almacenadas en la misma carpeta para su posterior uso desde otras aplicaciones.

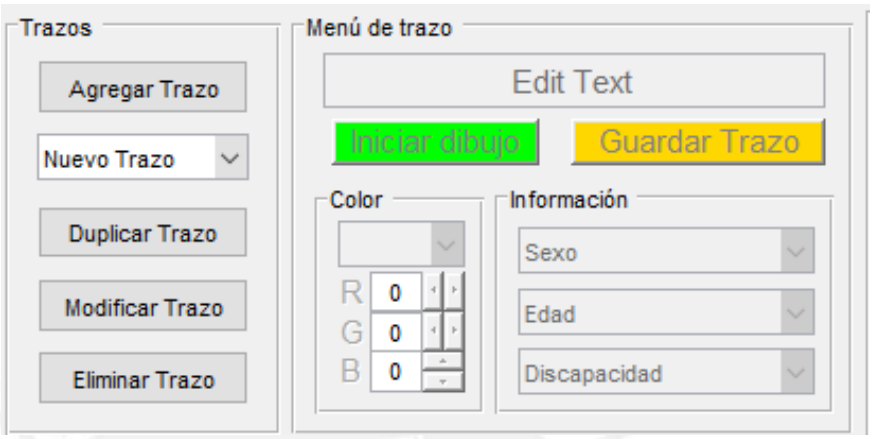


Importar croquis.

4. Trazos

4.1. Crear trazo

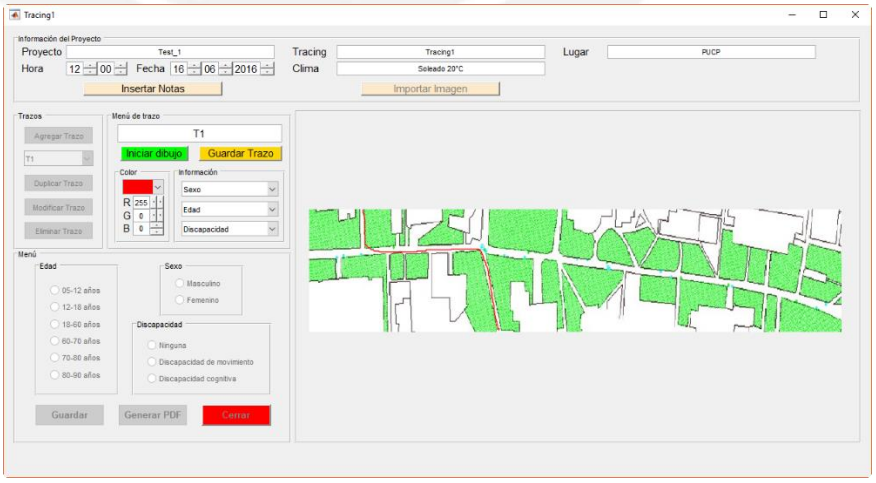
Para crear un trazo pulse el botón “Agregar Trazo”. El programa solicitará que ingrese el nombre de la nueva actividad. Solo se permite caracteres alfanuméricos en el nombre. Luego de crear la actividad se habilitará el panel “Menú de trazo”.



Panel Trazos.

4.2. Dibujar trazo

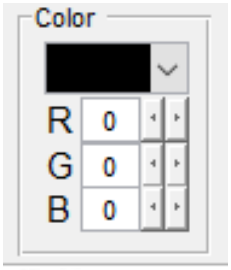
Los trazos se dibujan sobre el croquis. Para dibujar el trazo presione el botón Iniciar dibujo. Todos los elementos de la pantalla se deshabilitarán. Haga clic en el punto inicial del trazo y sin dejar de presionar el botón del mouse desplace el cursor mientras dibuja el trazo. Al soltar el botón el panel Menú de Trazo volverá a habilitarse. Si vuelve a presionar el botón Iniciar dibujo se borrará el trazo existente y podrá volver a dibujarlo.



Dibujar trazo.

4.3. Personalizar

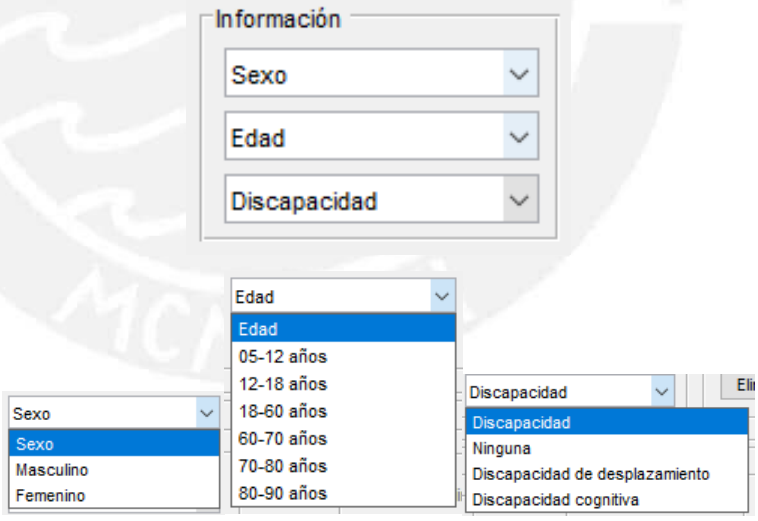
Para identificar los trazos puede asignarles un color particular. Se provee una serie de colores predefinidos que pueden ser seleccionados desde el menú desplegable en el subpanel Color. Sin embargo, estos no son los únicos colores disponibles ya que puede generar cualquier combinación variando las componentes RGB.



Personalizar trazo.

4.4. Información adicional

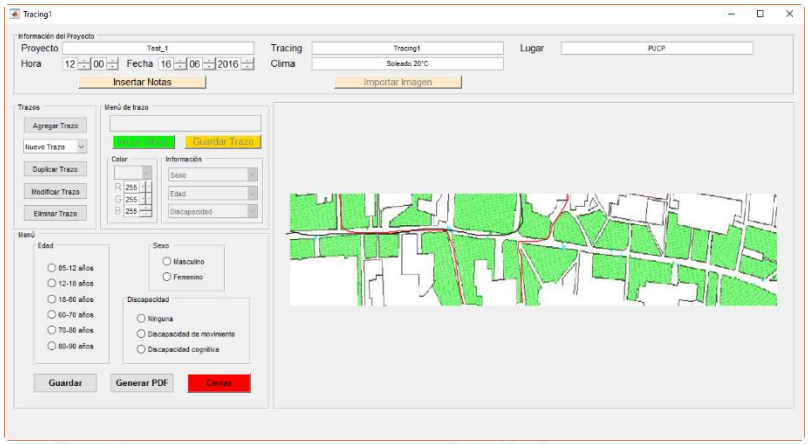
También puede asignar información adicional a los trazos ingresados. Los criterios asignables son sexo, rango de edad o la presencia de discapacidades de las personas registradas. Especificar esta información permite que el programa pueda elaborar reportes personalizados. Esta opción se verá más adelante.



Información adicional.

4.5. Guardar trazo

Luego de ingresar toda la información trazo presione el botón Guardar. El programa guardará la información ingresada y mostrará en el croquis todos los trazos existentes. El panel Menú de trazo se deshabilitará.



Guardar actividad.

4.6. Modificar/duplicar trazo

Si se desea editar un trazo existente debe seleccionarlo en el menú desplegable en el panel de Actividades y luego presionar el botón Modificar Trazo. Se habilitará el panel Actividad con la información seleccionada.

Además, también es posible duplicar un trazo. De esta manera se crea una copia de un trazo existente para preservar las propiedades del trazo original. Se solicitará un nuevo nombre para el trazo duplicado.

4.7. Eliminar trazo

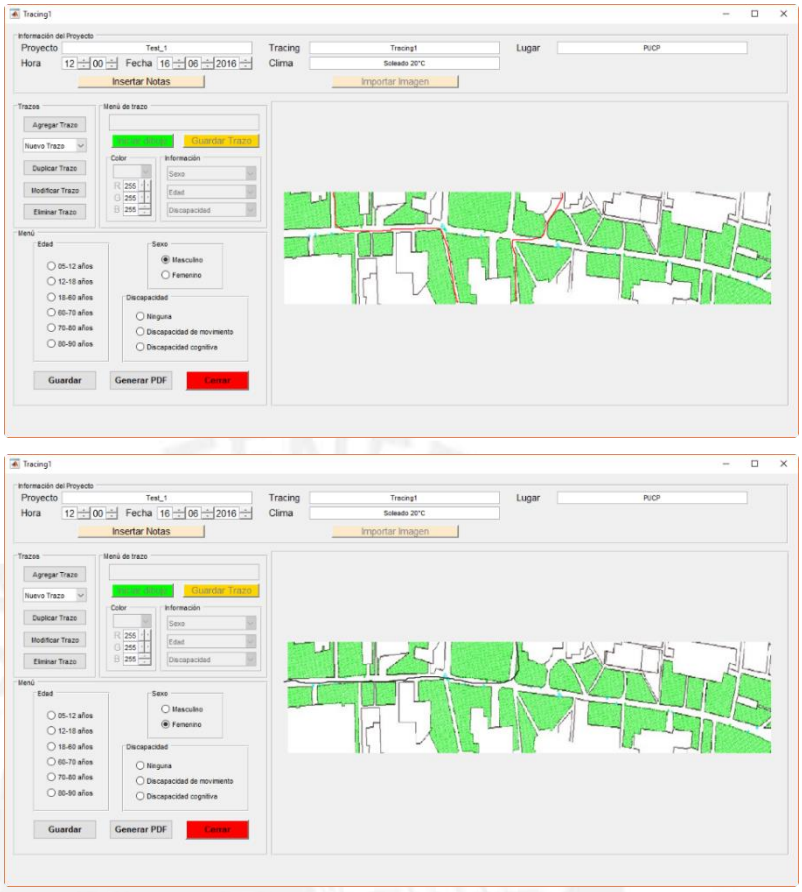
Para eliminar un trazo, selecciónelo en el menú desplegable y presione el botón Eliminar Trazo. Se solicitará la confirmación del usuario.

IMPORTANTE: Los trazos eliminados no podrán ser recuperados.

5. Visualización

En el panel inferior se cuenta con varios botones que permiten filtrar la información que se muestra en pantalla. Se mostrará en pantalla aquellos trazos que cumplan con los criterios de sexo, edad y presencia de discapacidades definidos.

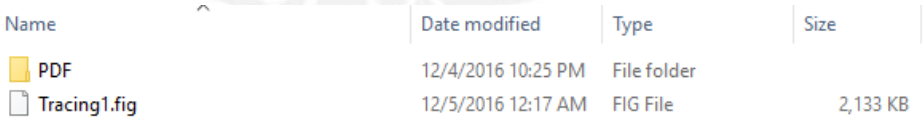
IMPORTANTE: En cada criterio, los estados de no marcar ningún botón o marcar todos son equivalentes.



Criterios de visualización.

6. Guardar archivo

Para guardar el archivo pulse el botón Guardar. El programa creará un archivo dentro de la carpeta asignada a la herramienta en la carpeta del proyecto. Este archivo no debe ser manipulado por el usuario.



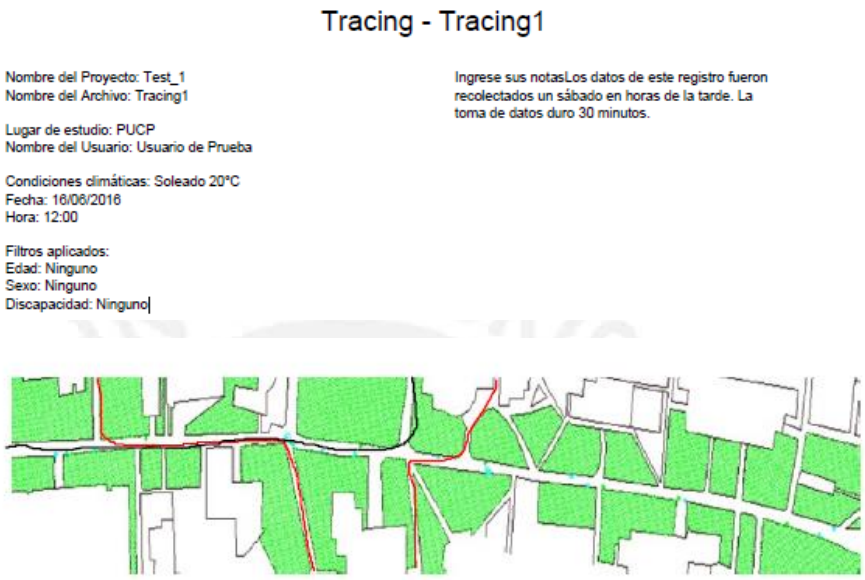
Guardar archivo.

7. Generar reportes

El programa permite generar reportes en formato PDF. Para ello basta con presionar el botón “Generar PDF”. El programa automáticamente generará un archivo en formato PDF dentro de la carpeta PDF que se muestra en la imagen

anterior. La información mostrada en el reporte coincide con los criterios de visualización activos.

El archivo se abrirá automáticamente luego de ser creado. El programa está diseñado para no sobrescribir los reportes generados, de esta manera se pueden generar reportes con distintas configuraciones de visualización.


Tracing - Tracing1

Nombre del Proyecto: Test_1
Nombre del Archivo: Tracing1

Lugar de estudio: PUCP
Nombre del Usuario: Usuario de Prueba

Condiciones climáticas: Soleado 20°C
Fecha: 16/06/2016
Hora: 12:00

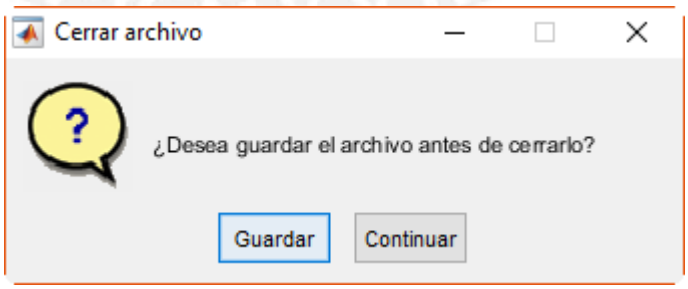
Filtros aplicados:
Edad: Ninguno
Sexo: Ninguno
Discapacidad: Ninguno

Ingrese sus notasLos datos de este registro fueron recolectados un sábado en horas de la tarde. La toma de datos duro 30 minutos.

Reporte PDF.

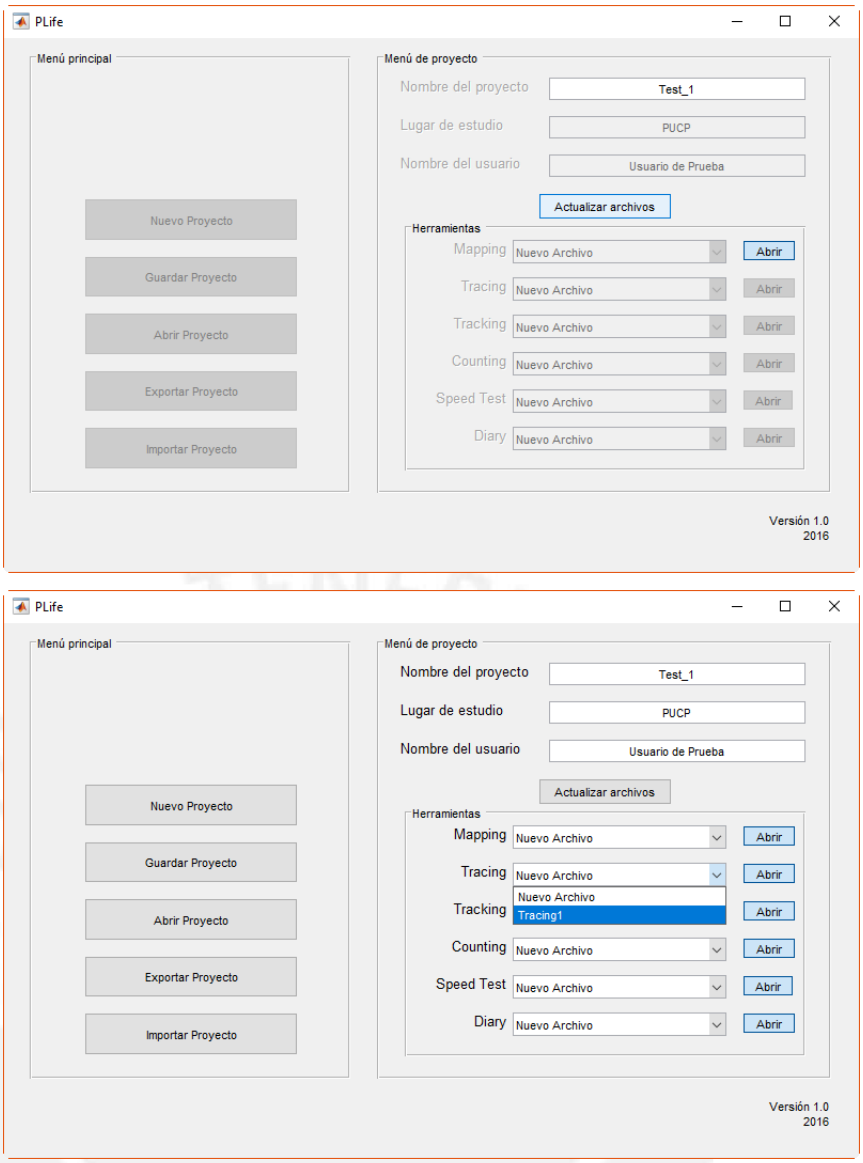
8. Cerrar herramienta

Finalmente, para regresar a la pantalla principal del programa presione el botón Cerrar. El programa consultará si desea guardar el archivo o si desea continuar sin guardar. Luego se cerrará la ventana de la herramienta y se abrirá la pantalla principal.



Cerrar herramienta.

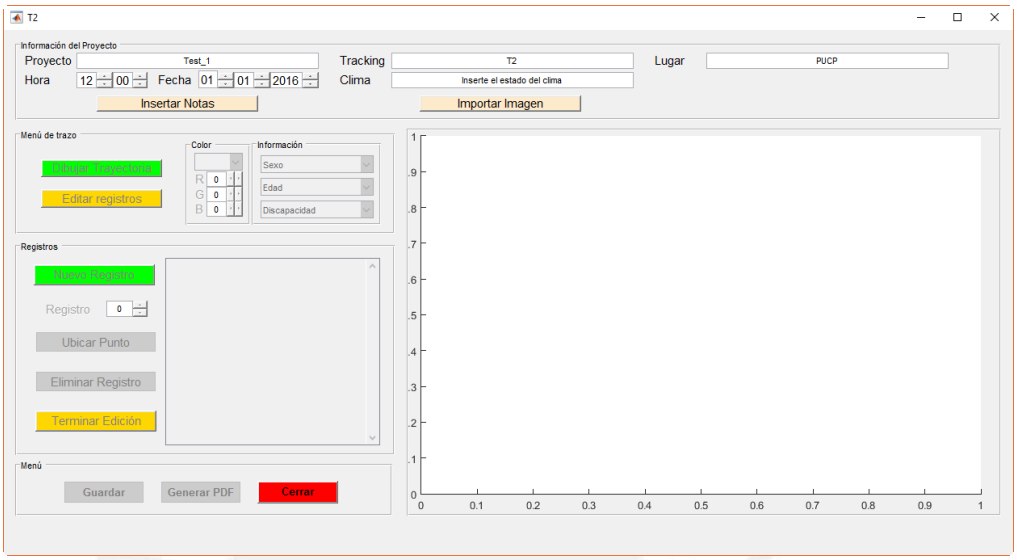
La ventana principal se abrirá con todos los elementos deshabilitados. Presione el botón Actualizar Archivos para habilitar la ventana. Si el archivo fue guardado, ahora este será visible en su respectivo menú desplegable.



Habilitar Pantalla Principal.

Capítulo 5 – Tracking

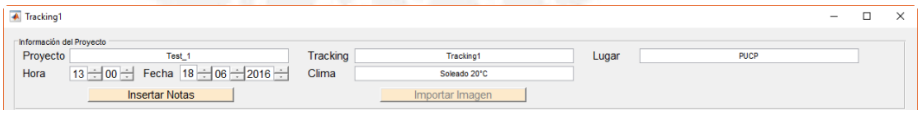
Al crear un nuevo archivo de la herramienta Tracking se abre la siguiente pantalla.



Pantalla inicial Tracking.

1. Editar información del proyecto

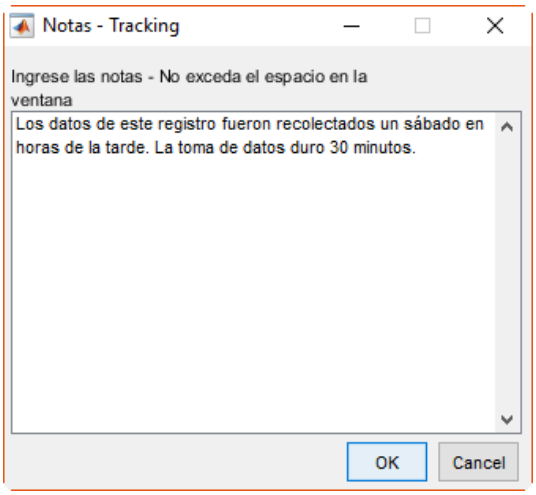
   En la parte superior se encuentra el panel Información del Proyecto. Dentro del panel hay cuatro campos de texto. El primero muestra el nombre del proyecto; el segundo, el nombre del archivo; y el tercero, el nombre del lugar analizado. Estos recuadros no son editables y muestran la información ingresada en la pantalla principal. El cuarto recuadro permite ingresar las condiciones climáticas registradas. En el mismo panel se cuenta con botones para definir la hora y fecha del registro. Toda la información ingresada se mostrará en el reporte generado por el programa.



Información del Proyecto.

2. Ingresar notas de texto.

   También se provee un espacio para ingresar notas de texto. Para ello se debe pulsar el botón Insertar Notas y se habilitará una ventana para ingresar el texto. El usuario debe tener cuidado de no exceder el espacio disponible en pantalla, caso contrario el texto no se mostrará apropiadamente en el reporte.

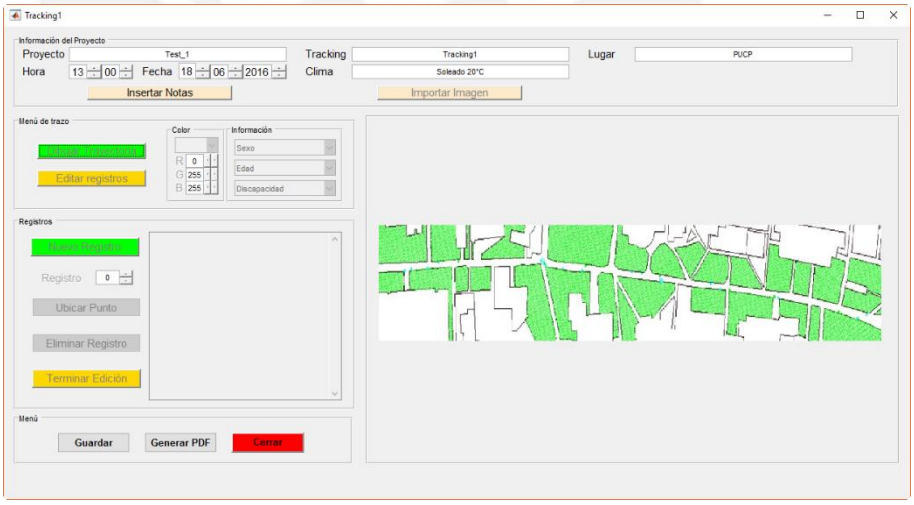


Notas de texto.

3. Importar croquis

Esta herramienta necesita que un croquis del espacio analizado sea importado. Para ello, el usuario debe contar con dicho croquis en un archivo en formato JPG. Para importar la imagen se debe presionar el botón “Importar Imagen”. Luego se habilita una ventana para seleccionar el archivo deseado. Luego de seleccionar la imagen esta se muestra en la pantalla de la aplicación.

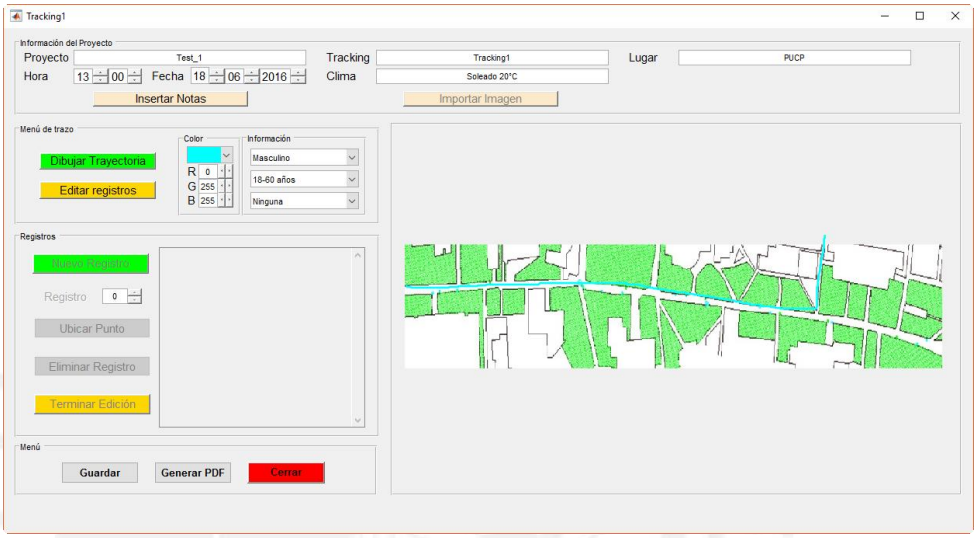
IMPORTANTE: Por defecto se muestra la carpeta Imágenes dentro del proyecto. Si el usuario selecciona un archivo que no esté dentro de esta carpeta el programa creará una copia de la imagen en la carpeta “Imágenes”. De esta manera, todas las imágenes del proyecto son almacenadas en la misma carpeta para su posterior uso desde otras aplicaciones.



Importar croquis.

4. Dibujar trayectoria

Presione el botón Dibujar Trayectoria. Todos los elementos de la pantalla se deshabilitarán. Sobre el croquis, haga clic en el punto inicial del trazo y sin dejar de presionar el botón del mouse desplace el cursor mientras dibuja el trazo. Al soltar el botón el panel Menú de Trazo volverá a habilitarse. Si vuelve a presionar el botón Iniciar dibujo se borrará el trazo existente y podrá volver a dibujarlo.

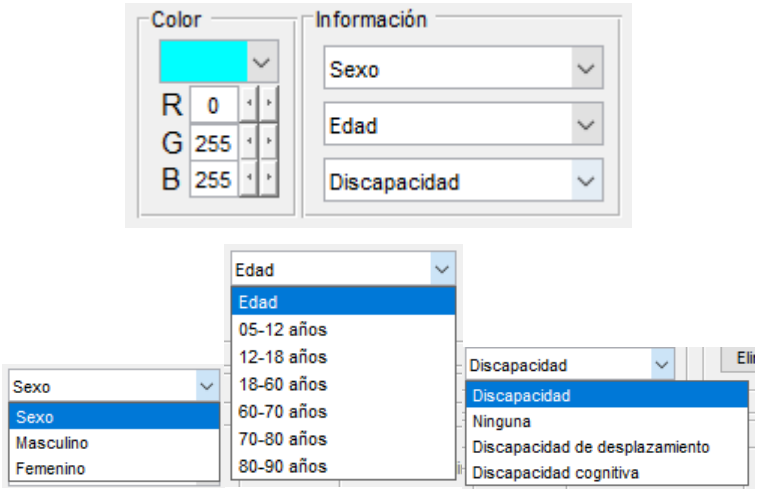


Panel Trazos.

5. Color e información adicional

El programa permite asignar un color personalizado al trazo para contrastar con el fondo cargado. Se provee una serie de colores predefinidos que pueden ser seleccionados desde el menú desplegable en el subpanel Color. Sin embargo, estos no son los únicos colores disponibles ya que puede generar cualquier combinación variando las componentes RGB.

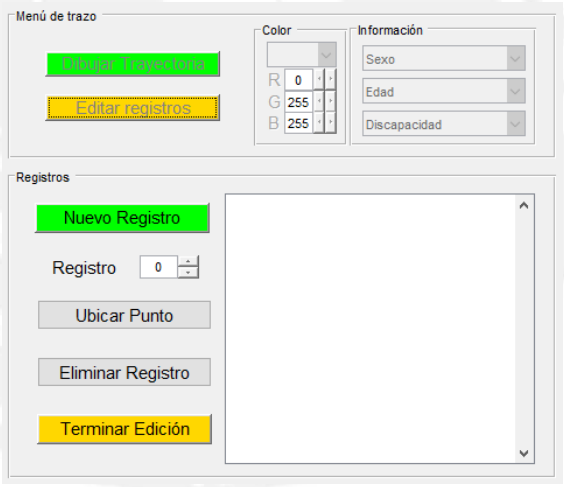
También puede asignar información adicional según la información recolectada. Los criterios asignables son sexo, rango de edad o la presencia de discapacidades de las personas registradas.



Información adicional.

6. Registros

El objetivo de esta herramienta es registrar las acciones que realiza la persona estudiada. Para ello el programa permite ingresar múltiples registros y ubicarlos espacialmente en la ruta. Para ello debe presionar el botón Editar registros tras lo cual se habilitará el respectivo panel.



Panel de Registros.

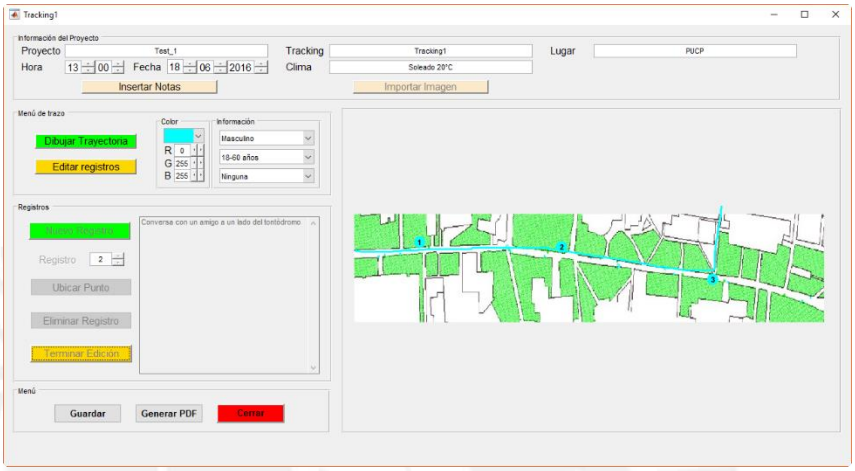
Para añadir un registro presione el botón Nuevo Registro. Se añadirá un número al contador que se encuentra en el panel de Registros. Puede desplazarse entre los registros existentes y el programa le mostrará la información del registro activo.

En el campo de la derecha puede añadir las notas de texto libremente. Para ubicar el lugar donde ocurrió el registro active el botón Ubicar Punto. Se desactivarán todos los elementos de la pantalla hasta que haga clic sobre el

croquis y se añada el punto del registro activo. Cada punto estará etiquetado con el número del registro que le corresponde. También puede borrar los registros existentes. Luego de borrar un registro la numeración será reasignada.

IMPORTANTE: Los registros eliminados no podrán ser recuperados.

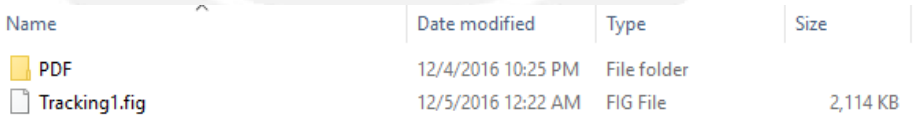
Pulse el botón Terminar Edición para volver al estado anterior. Todos los puntos se muestran en el croquis.



Guardar actividad.

7. Guardar archivo

Para guardar el archivo pulse el botón Guardar. El programa creará un archivo dentro de la carpeta asignada a la herramienta en la carpeta del proyecto. Este archivo no debe ser manipulado por el usuario.



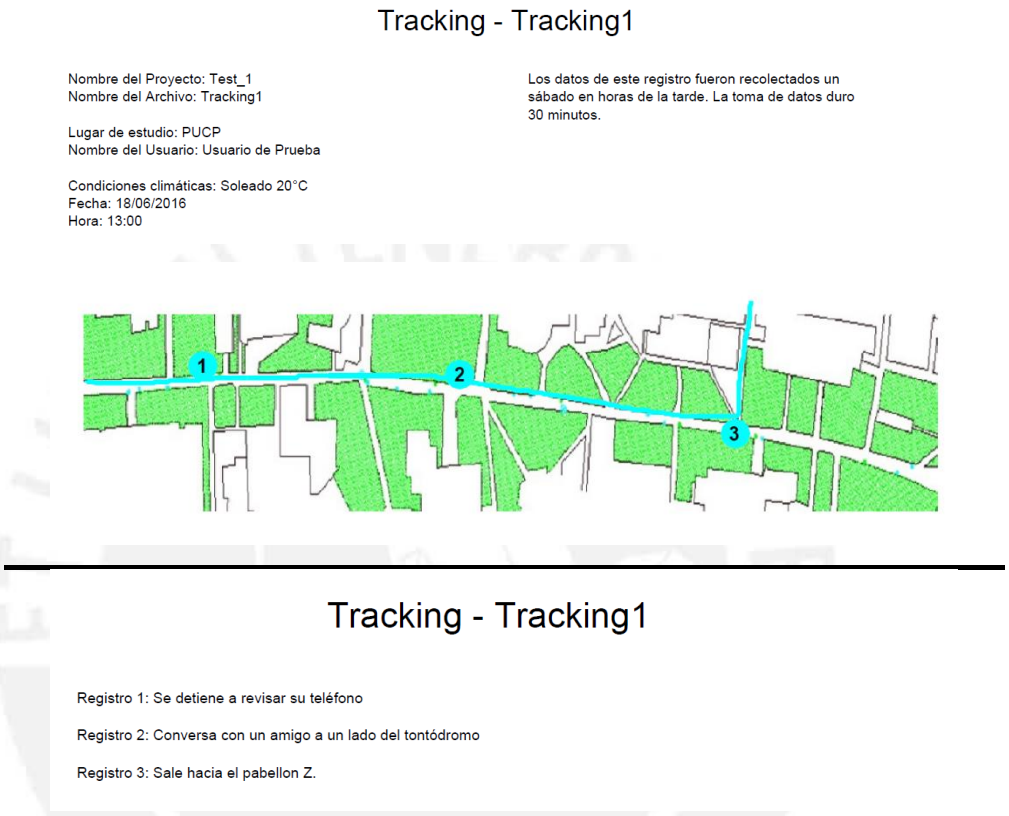
Guardar archivo.

8. Generar reportes

El programa permite generar reportes en formato PDF. Para ello basta con presionar el botón “Generar PDF”. El programa automáticamente generará un archivo en formato PDF dentro de la carpeta PDF que se muestra en la imagen anterior. La información mostrada en el reporte coincide con los criterios de visualización activos.

Cada reporte se compone de dos páginas. La primera muestra los datos generales del proyecto y el croquis. En la segunda se muestra las notas de texto de los registros ingresados.

El archivo se abrirá automáticamente luego de ser creado. El programa está diseñado para no sobrescribir los reportes generados, de esta manera se pueden generar reportes con distintas configuraciones de visualización.


Tracking - Tracking1

Nombre del Proyecto: Test_1
Nombre del Archivo: Tracking1

Lugar de estudio: PUCP
Nombre del Usuario: Usuario de Prueba

Condiciones climáticas: Soleado 20°C
Fecha: 18/06/2016
Hora: 13:00

Los datos de este registro fueron recolectados un sábado en horas de la tarde. La toma de datos duro 30 minutos.

Tracking - Tracking1

Registro 1: Se detiene a revisar su teléfono

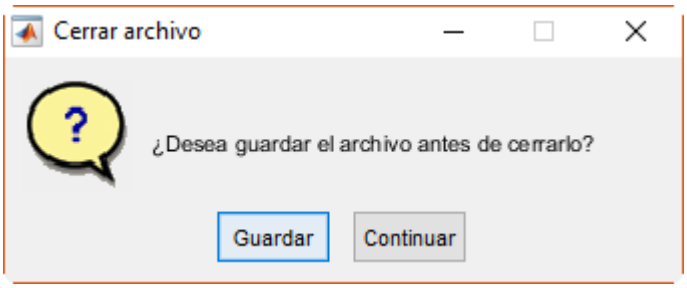
Registro 2: Conversa con un amigo a un lado del tontódromo

Registro 3: Sale hacia el pabellon Z.

Reporte PDF.

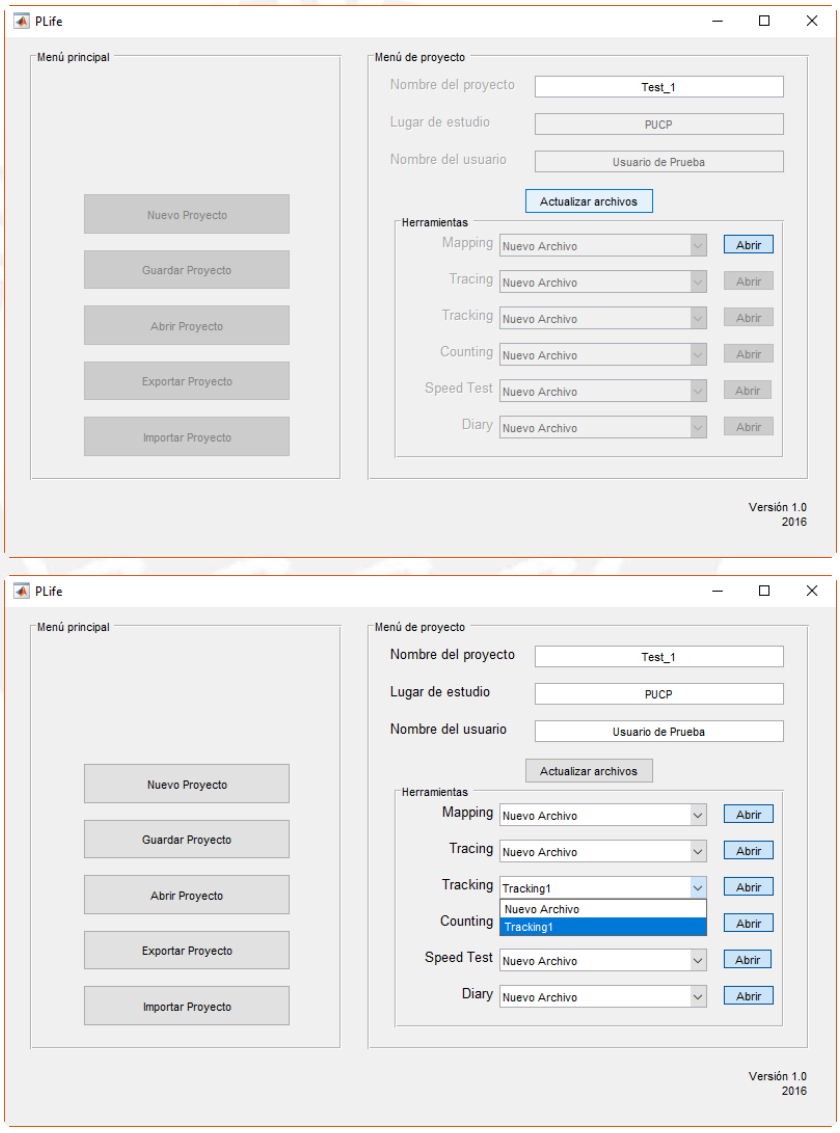
9. Cerrar herramienta

Finalmente, para regresar a la pantalla principal del programa presione el botón Cerrar. El programa consultará si desea guardar el archivo o si desea continuar sin guardar. Luego se cerrará la ventana de la herramienta y se abrirá la pantalla principal.



Cerrar herramienta.

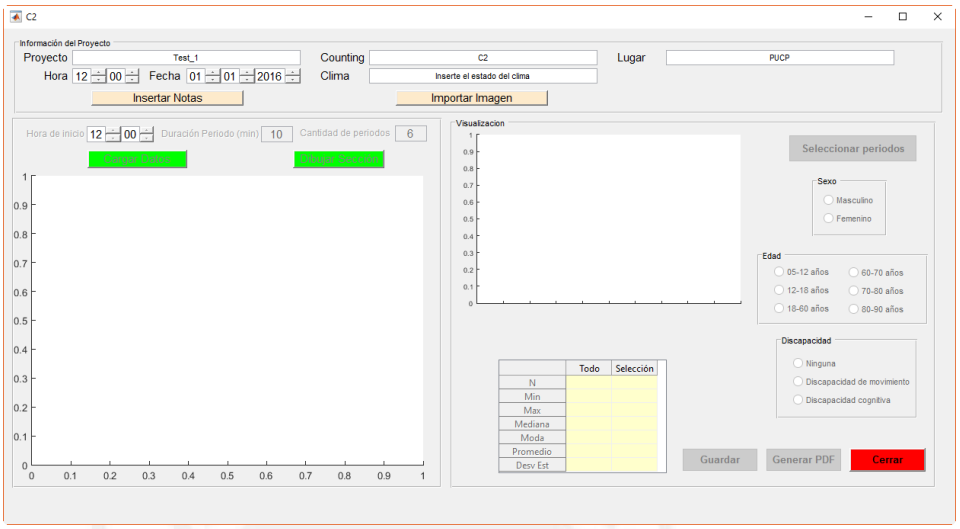
La ventana principal se abrirá con todos los elementos deshabilitados. Presione el botón Actualizar Archivos para habilitar la ventana. Si el archivo fue guardado, ahora este será visible en su respectivo menú desplegable.



Habilitar Pantalla Principal.

Capítulo 6 – Counting

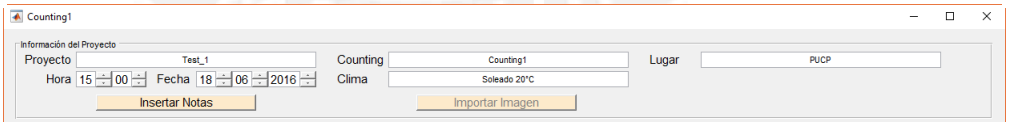
Al crear un nuevo archivo de la herramienta Counting se abre la siguiente pantalla.



Pantalla inicial Tracking.

1. Editar información del proyecto

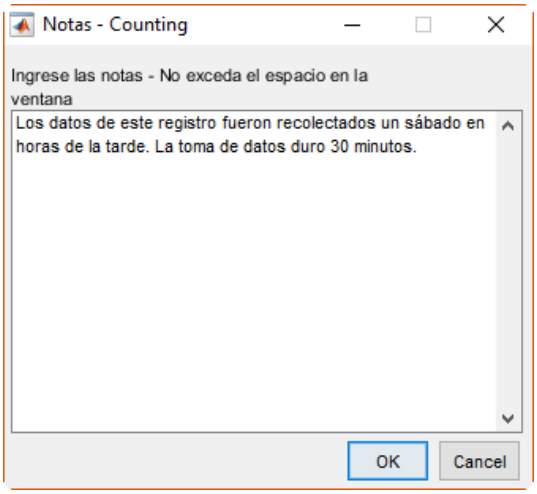
   En la parte superior se encuentra el panel Información del Proyecto. Dentro del panel hay cuatro campos de texto. El primero muestra el nombre del proyecto; el segundo, el nombre del archivo; y el tercero, el nombre del lugar analizado. Estos recuadros no son editables y muestran la información ingresada en la pantalla principal. El cuarto recuadro permite ingresar las condiciones climáticas registradas. En el mismo panel se cuenta con botones para definir la hora y fecha del registro. Toda la información ingresada se mostrará en el reporte generado por el programa.



Información del Proyecto.

2. Ingresar notas de texto.

   También se provee un espacio para ingresar notas de texto. Para ello se debe pulsar el botón Insertar Notas y se habilitará una ventana para ingresar el texto. El usuario debe tener cuidado de no exceder el espacio disponible en pantalla, caso contrario el texto no se mostrará apropiadamente en el reporte.

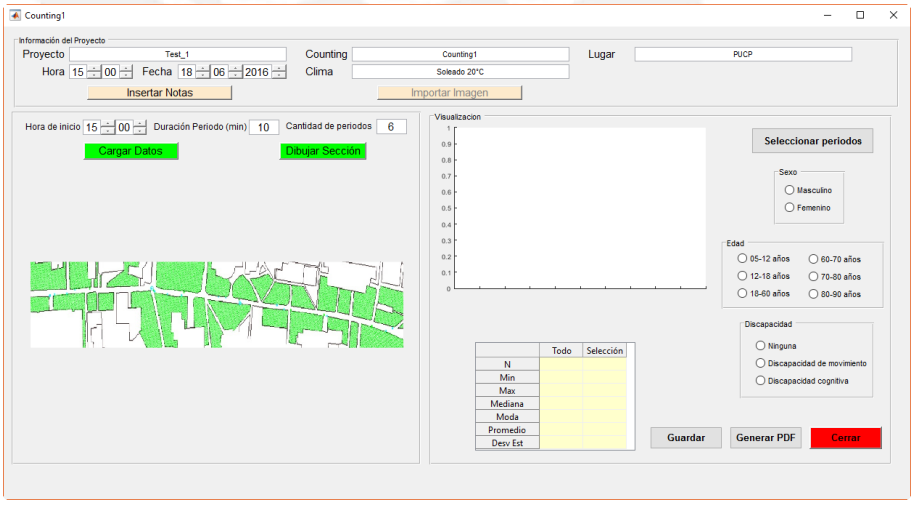


Notas de texto.

3. Importar croquis

Esta herramienta necesita que un croquis del espacio analizado sea importado. Para ello, el usuario debe contar con dicho croquis en un archivo en formato JPG. Para importar la imagen se debe presionar el botón “Importar Imagen”. Luego se habilita una ventana para seleccionar el archivo deseado. Luego de seleccionar la imagen esta se muestra en la pantalla de la aplicación.

IMPORTANTE: Por defecto se muestra la carpeta Imágenes dentro del proyecto. Si el usuario selecciona un archivo que no esté dentro de esta carpeta el programa creará una copia de la imagen en la carpeta “Imágenes”. De esta manera, todas las imágenes del proyecto son almacenadas en la misma carpeta para su posterior uso desde otras aplicaciones.



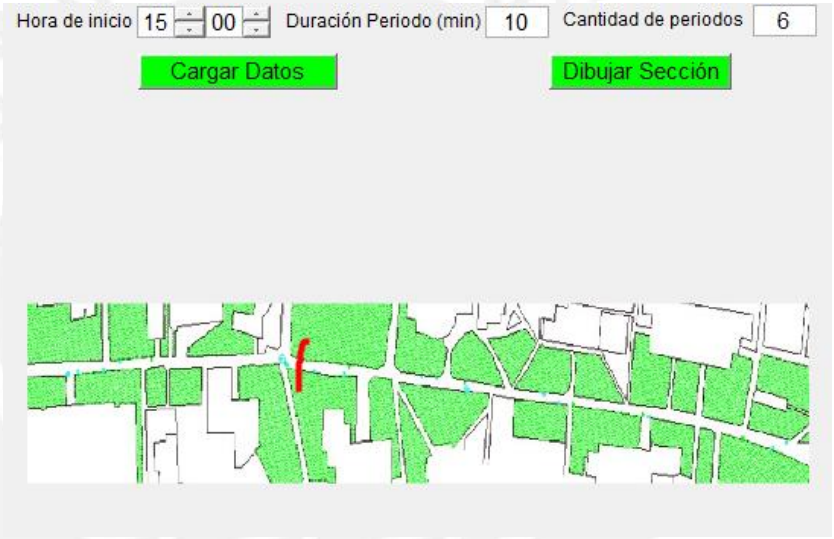
Importar croquis.

4. Definir parámetros del registro

La herramienta Counting permite registrar flujos peatonales. Para ello es necesario definir los parámetros del registro.

En primer lugar, se debe indicar la sección en la cual se registraron los datos. Para ello, presione el botón Dibujar sección y el programa deshabilitará todos los elementos de la pantalla hasta que dibuje, haciendo uso del cursor, un trazo sobre el croquis.

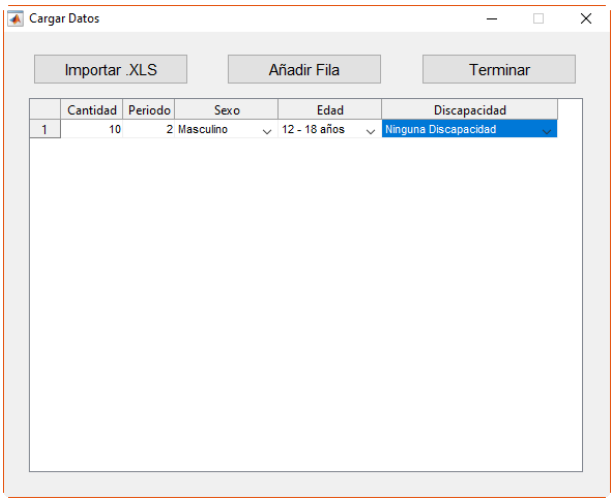
En segundo lugar, es necesario definir el intervalo temporal en el cual se registraron los datos. Se acostumbra dividir el periodo en intervalos iguales. Para ello, se define una hora de inicio, la cantidad de intervalos de tiempo y la duración, en minutos, de cada uno.



Parámetros del registro

5. Ingreso de datos

Para ingresar los datos presione el botón Cargar Datos. Se abrirá una ventana como la que se muestra a continuación.

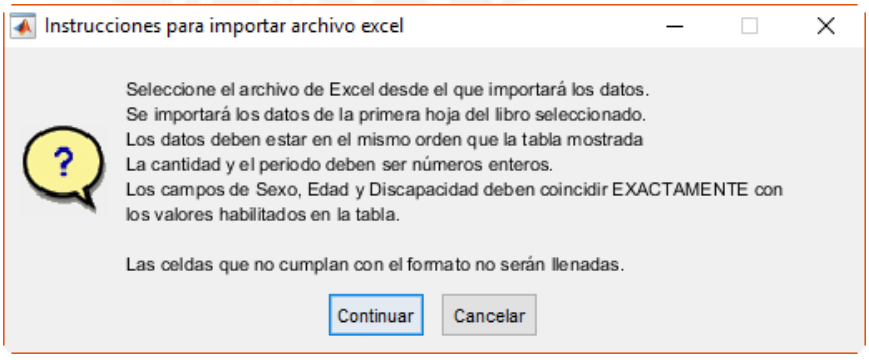


Pantalla para cargar datos

En la primera columna, cantidad, se indica las personas contabilizadas. En la segunda columna, periodo, se especifica el periodo temporal en el cual se registró el conteo. Las tres columnas restantes son opcionales y permiten especificar características de los peatones registrados como sexo, rango de edad o la presencia de discapacidades.

Use el botón Añadir fila para aumentar el tamaño de la tabla e ingresar los datos manualmente. No es necesario que los datos estén ordenados por periodo. Tampoco es necesario rellenar todas las filas. Aquellas filas que no tengan datos válidos en la primera y segunda columna serán obviadas al momento de cargar la información.

El programa también permite importar datos de una tabla de Excel. Para usar esta funcionalidad la tabla debe seguir el formato especificado por el programa al presionar el botón Importar .XLS.



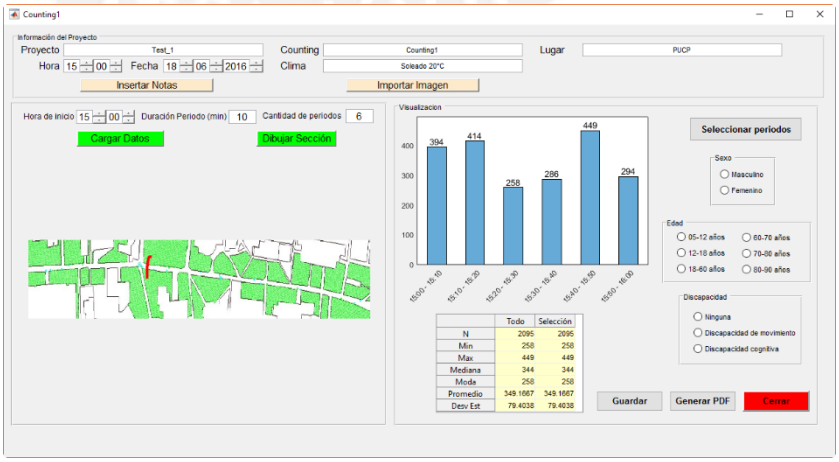
Pantalla para cargar datos

- Los datos en la tabla de Excel deben encontrarse en la primera hoja y comenzar en la primera fila, sin encabezado.
- Las columnas deben estar ordenadas igual que la tabla que se muestra en el programa.
- Los valores numéricos deben ser números enteros.
- Las características de sexo, edad y presencia de discapacidades deben coincidir EXACTAMENTE con los valores disponibles en la tabla. Si algún campo no coincide con los textos válidos la celda quedará vacía.
- Junto con esta guía se provee un archivo de ejemplo para ilustrar esta funcionalidad.

	A	B	C	D	E
1	19	5	Masculino	12 - 18 años	Ninguna Discapacidad
2	15	1	Femenino	12 - 18 años	Ninguna Discapacidad
3	23	1	Masculino	18 - 60 años	Ninguna Discapacidad
4	16	2	Femenino	12 - 18 años	Ninguna Discapacidad
5	29	5	Masculino	05 - 12 años	Ninguna Discapacidad
6	24	6	Masculino	12 - 18 años	Ninguna Discapacidad
7	30	4		18 - 60 años	Ninguna Discapacidad
8	22	1	Femenino	12 - 18 años	Ninguna Discapacidad
9	16	4	Masculino	05 - 12 años	
10	18	4	Masculino	12 - 18 años	Ninguna Discapacidad
11	28	1	Femenino	12 - 18 años	Ninguna Discapacidad
12	28	2	Femenino	18 - 60 años	Ninguna Discapacidad
13	13	1	Masculino	12 - 18 años	Ninguna Discapacidad
14	19	4	Masculino	12 - 18 años	Ninguna Discapacidad

Ejemplo para cargar datos de Excel.

Luego de cargar los datos estos se mostrarán en la tabla del programa. Para finalizar el ingreso de datos presione el botón Terminar. La ventana se cerrará y en la pantalla principal, si ingreso al menos un dato válido, se mostrará un histograma y una tabla resumen de la información ingresada.



Resultados del programa.

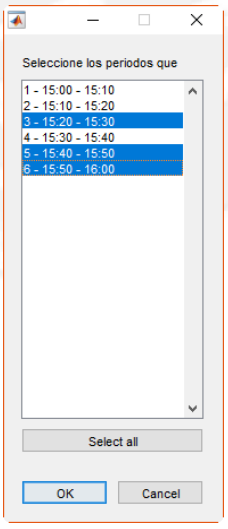
Si desea añadir nuevos datos al registro vuelva a presionar el botón Cargar datos. La tabla mostrará los datos existentes, a los cuales puede añadir nuevas filas manualmente. Si vuelve a cargar un archivo de Excel, los datos serán añadidos al final de la tabla actual.

IMPORTANTE: Tenga en cuenta que el programa no ha implementado una funcionalidad para eliminar filas. Para eliminar una fila debe dejar en blanco las celdas numéricas de manera que, al terminar la carga de datos, el programa obvie estas filas y no se muestren la próxima vez que se carguen datos.

6. Visualización

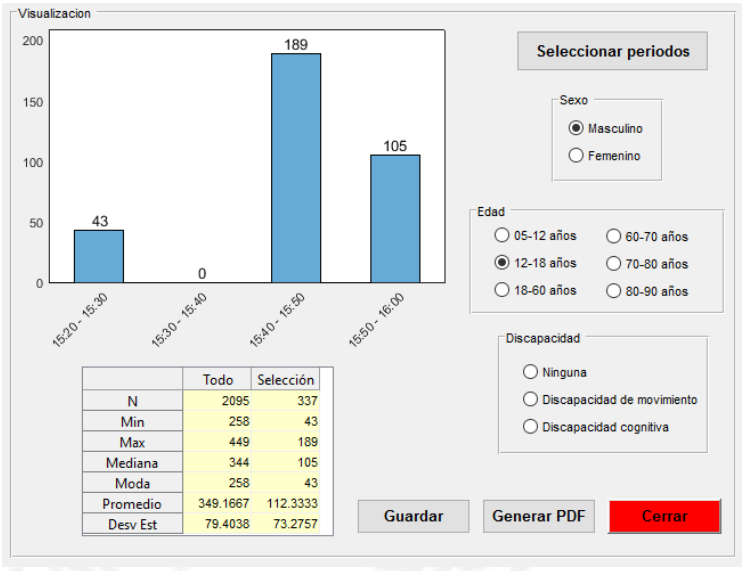
En el panel derecho se cuenta con varios botones que permiten filtrar la información que se muestra en pantalla. Se mostrará en el histograma aquellos datos que cumplan con los criterios de sexo, edad y presencia de discapacidades definidos.

Además de los criterios mencionados también es posible mostrar los datos de algunos periodos de tiempo. Para ello presione el botón Seleccionar periodos. Se habilitará una ventana en donde debe seleccionar los intervalos que se mostrarán. Para seleccionar múltiples periodos mantenga el botón Ctrl presionado.



Seleccionar los periodos.

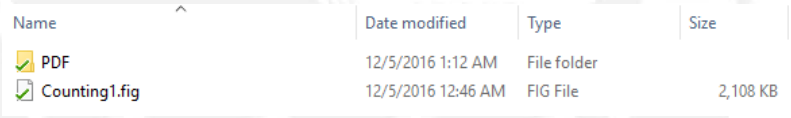
IMPORTANTE: En cada criterio, los estados de no marcar ningún botón o marcar todos son equivalentes.



Criterios de visualización.

7. Guardar archivo

Para guardar el archivo pulse el botón Guardar. El programa creará un archivo dentro de la carpeta asignada a la herramienta en la carpeta del proyecto. Este archivo no debe ser manipulado por el usuario.

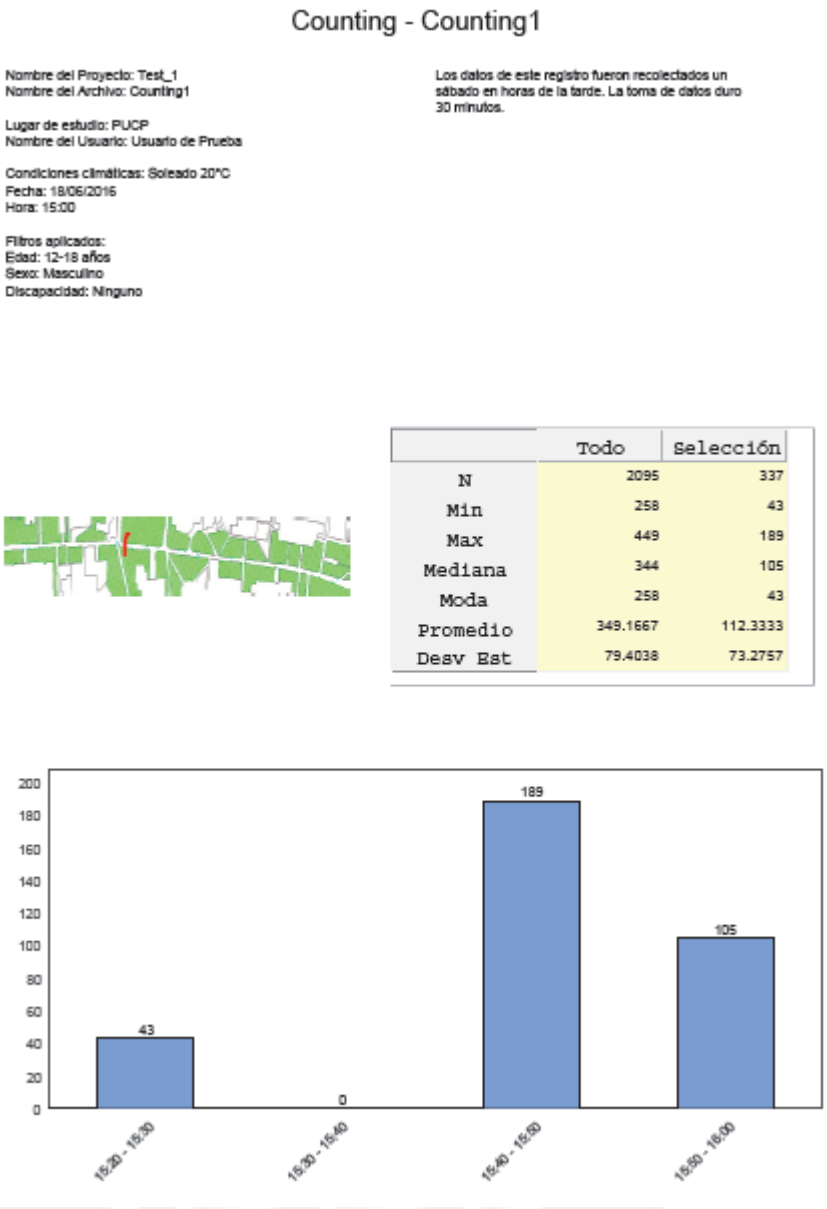


Guardar archivo.

8. Generar reportes

El programa permite generar reportes en formato PDF. Para ello basta con presionar el botón “Generar PDF”. El programa automáticamente generará un archivo en formato PDF dentro de la carpeta PDF que se muestra en la imagen anterior. La información mostrada en el reporte coincide con los criterios de visualización activos.

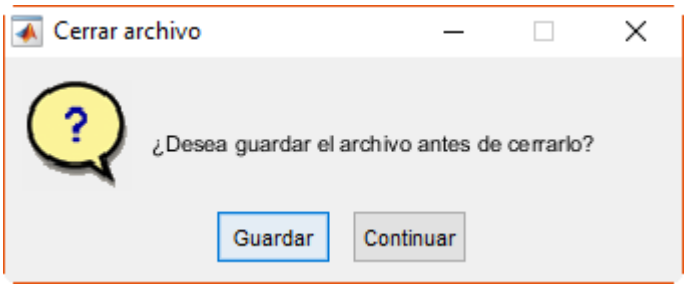
El archivo se abrirá automáticamente luego de ser creado. El programa está diseñado para no sobrescribir los reportes generados, de esta manera se pueden generar reportes con distintas configuraciones de visualización.



Reporte PDF.

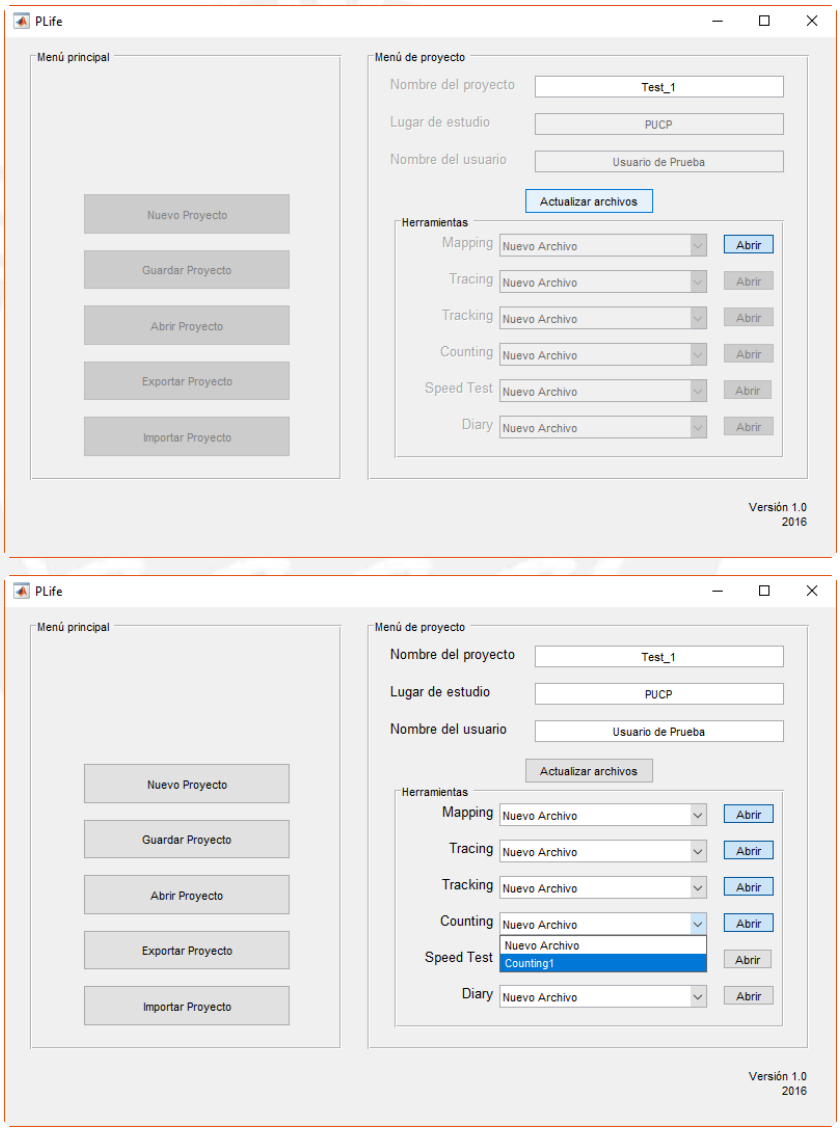
9. Cerrar herramienta

Finalmente, para regresar a la pantalla principal del programa presione el botón Cerrar. El programa consultará si desea guardar el archivo o si desea continuar sin guardar. Luego se cerrará la ventana de la herramienta y se abrirá la pantalla principal.



Cerrar herramienta.

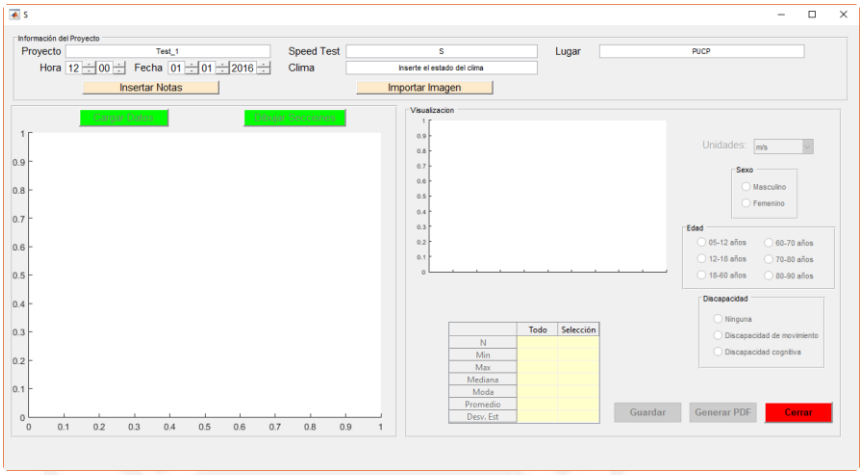
La ventana principal se abrirá con todos los elementos deshabilitados. Presione el botón Actualizar Archivos para habilitar la ventana. Si el archivo fue guardado, ahora este será visible en su respectivo menú desplegable.



Habilitar Pantalla Principal.

Capítulo 7 – Speed Test

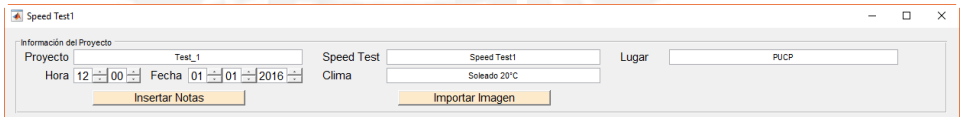
Al crear un nuevo archivo de la herramienta Speed Test se abre la siguiente pantalla.



Pantalla inicial Tracking.

1. Editar información del proyecto

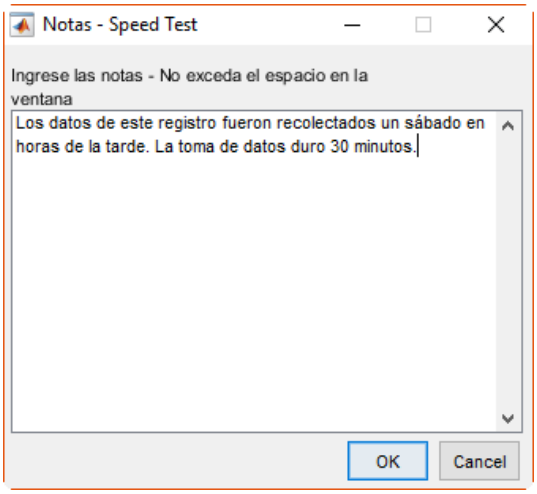
   En la parte superior se encuentra el panel Información del Proyecto. Dentro del panel hay cuatro campos de texto. El primero muestra el nombre del proyecto; el segundo, el nombre del archivo; y el tercero, el nombre del lugar analizado. Estos recuadros no son editables y muestran la información ingresada en la pantalla principal. El cuarto recuadro permite ingresar las condiciones climáticas registradas. En el mismo panel se cuenta con botones para definir la hora y fecha del registro. Toda la información ingresada se mostrará en el reporte generado por el programa.



Información del Proyecto.

2. Ingresar notas de texto.

   También se provee un espacio para ingresar notas de texto. Para ello se debe pulsar el botón Insertar Notas y se habilitará una ventana para ingresar el texto. El usuario debe tener cuidado de no exceder el espacio disponible en pantalla, caso contrario el texto no se mostrará apropiadamente en el reporte.

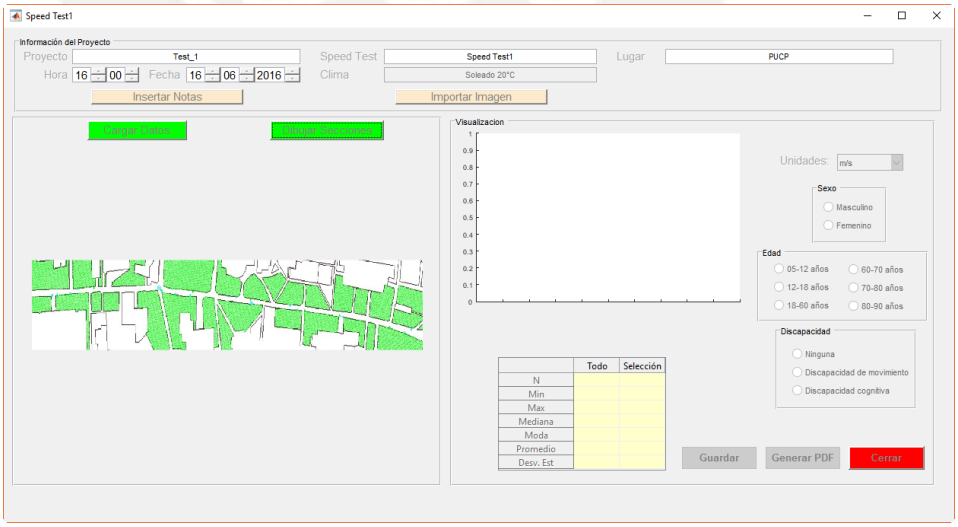


Notas de texto.

3. Importar croquis

Esta herramienta necesita que un croquis del espacio analizado sea importado. Para ello, el usuario debe contar con dicho croquis en un archivo en formato JPG. Para importar la imagen se debe presionar el botón “Importar Imagen”. Luego se habilita una ventana para seleccionar el archivo deseado. Luego de seleccionar la imagen esta se muestra en la pantalla de la aplicación.

IMPORTANTE: Por defecto se muestra la carpeta Imágenes dentro del proyecto. Si el usuario selecciona un archivo que no esté dentro de esta carpeta el programa creará una copia de la imagen en la carpeta “Imágenes”. De esta manera, todas las imágenes del proyecto son almacenadas en la misma carpeta para su posterior uso desde otras aplicaciones.

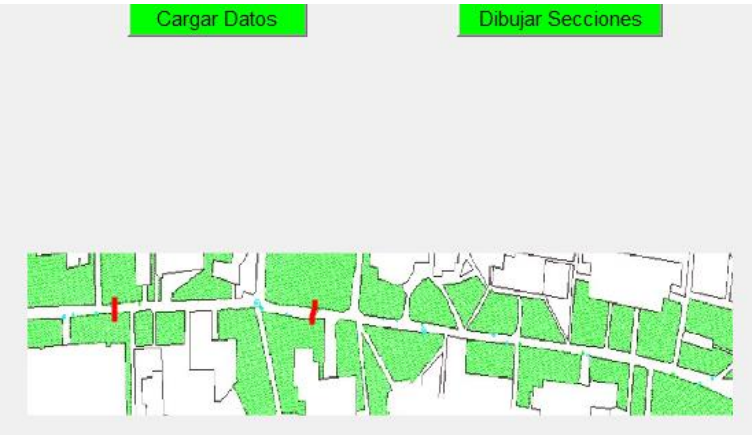


Importar croquis.

4. Definir parámetros del registro

La herramienta Speed Test permite registrar los tiempos de viaje de peatones para estimar su velocidad promedio. Para ello es necesario definir los parámetros del registro.

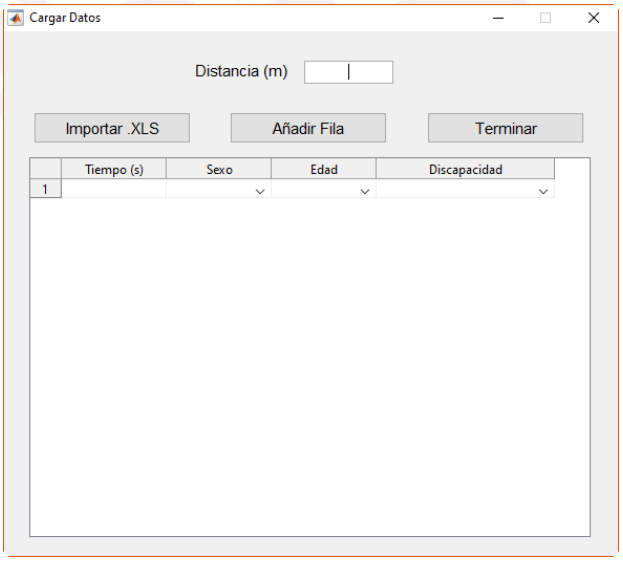
Se debe indicar las secciones en las cuales se registraron los datos. Para ello, presione el botón Dibujar sección y el programa deshabilitará todos los elementos de la pantalla hasta que dibuje, haciendo uso del cursor, dos trazos sobre el croquis.



Parámetros del registro

5. Ingreso de datos

Para ingresar los datos presione el botón Cargar Datos. Se abrirá una ventana como la que se muestra a continuación.



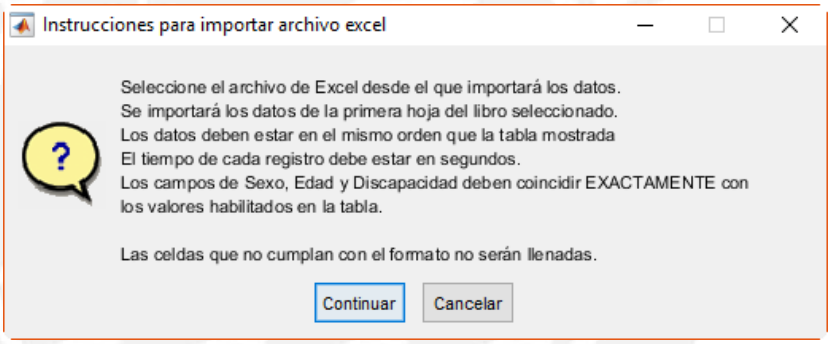
Pantalla para cargar datos

En la parte superior se debe indicar la distancia, en metros, que separa las secciones analizadas. Este dato será usado para calcular las velocidades.

En la primera columna se indica el tiempo, en segundos, registrado. Las tres columnas restantes son opcionales y permiten especificar características de los peatones registrados como sexo, rango de edad o la presencia de discapacidades.

Use el botón Añadir fila para aumentar el tamaño de la tabla e ingresar los datos manualmente. No es necesario rellenar todas las filas. Aquellas filas que no tengan datos válidos en la primera columna serán obviadas al momento de cargar la información.

El programa también permite importar datos de una tabla de Excel. Para usar esta funcionalidad la tabla debe seguir el formato especificado por el programa al presionar el botón Importar .XLS.



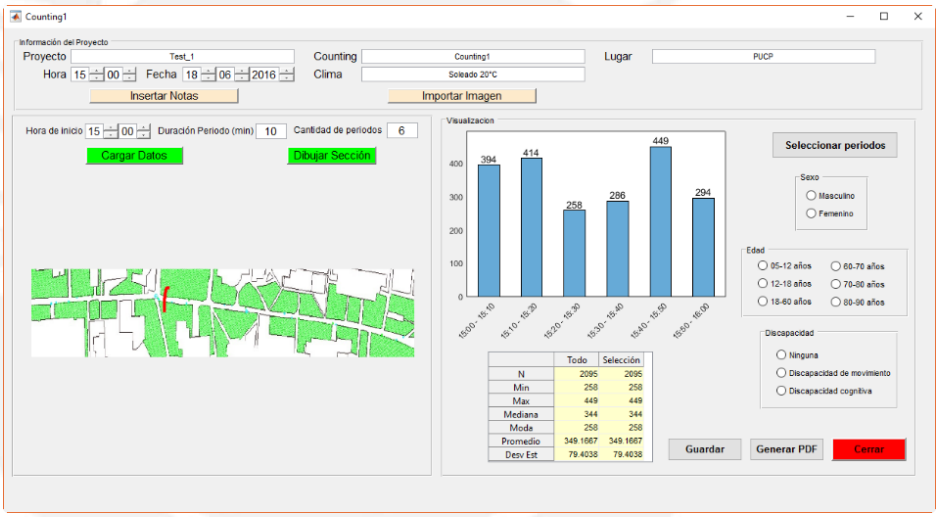
Pantalla para cargar datos

- Los datos en la tabla de Excel deben encontrarse en la primera hoja y comenzar en la primera fila, sin encabezado.
- Las columnas deben estar ordenada igual que la tabla que se muestra en el programa.
- Los tiempos deben estar en segundos.
- Las características de sexo, edad y presencia de discapacidades deben coincidir EXACTAMENTE con los valores disponibles en la tabla. Si algún campo no coincide con los textos válidos la celda quedará vacía.
- Junto con esta guía se provee un archivo de ejemplo para ilustrar esta funcionalidad.

	A	B	C	D
1	19.3	Masculino	12 - 18 años	Ninguna Discapacidad
2	15.4	Femenino	12 - 18 años	Ninguna Discapacidad
3	23.5	Masculino	18 - 60 años	Ninguna Discapacidad
4	16.1	Femenino	12 - 18 años	Ninguna Discapacidad
5	29.6	Masculino	05 - 12 años	Ninguna Discapacidad
6	24.1	Masculino	12 - 18 años	Ninguna Discapacidad
7	31.1		18 - 60 años	Ninguna Discapacidad
8	22.5	Femenino	12 - 18 años	Ninguna Discapacidad
9	21.1	Masculino	05 - 12 años	
10	18.8	Masculino	12 - 18 años	Ninguna Discapacidad
11	28.4	Femenino	12 - 18 años	Ninguna Discapacidad
12	28.3	Femenino	18 - 60 años	Ninguna Discapacidad
13	13.2	Masculino	12 - 18 años	Ninguna Discapacidad
14	19.5	Masculino	12 - 18 años	Ninguna Discapacidad

Ejemplo para cargar datos de Excel.

Luego de cargar los datos estos se mostrarán en la tabla del programa. Para finalizar el ingreso de datos presione el botón Terminar. La ventana se cerrará y en la pantalla principal, si ingreso al menos un dato válido, se mostrará un histograma y una tabla resumen de la información ingresada.



Resultados del programa.

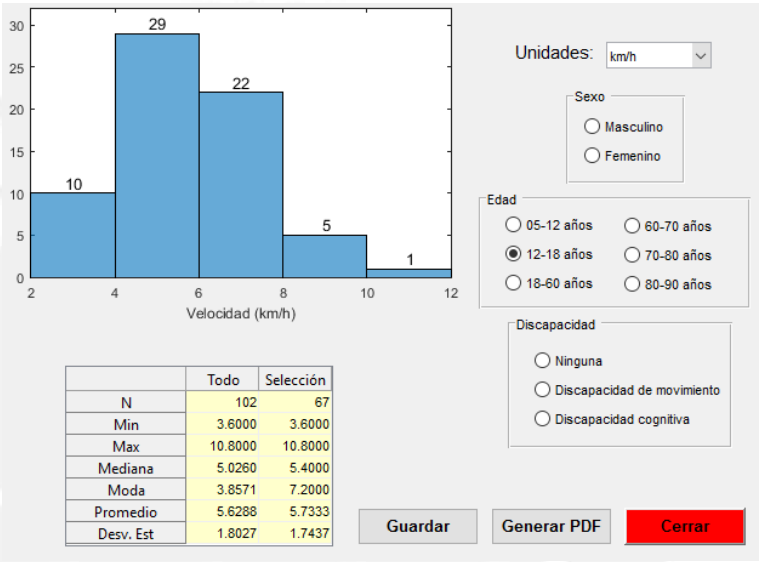
Si desea añadir nuevos datos al registro vuelva a presionar el botón Cargar datos. La tabla mostrará los datos existentes, a los cuales puede añadir nuevas filas manualmente. Si vuelve a cargar un archivo de Excel, los datos serán añadidos al final de la tabla actual.

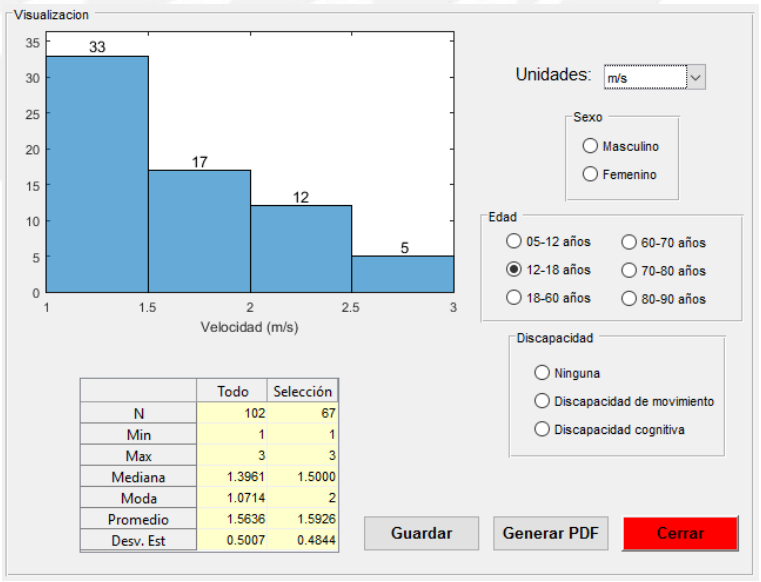
IMPORTANTE: Tenga en cuenta que el programa no ha implementado una funcionalidad para eliminar filas. Para eliminar una fila debe dejar en blanco las celdas numéricas de manera que, al terminar la carga de datos, el programa obvie estas filas y no se muestren la próxima vez que se carguen datos.

6. Visualización

En el panel derecho se cuenta con varios botones que permiten filtrar la información que se muestra en pantalla. Se mostrará en el histograma aquellos datos que cumplan con los criterios de sexo, edad y presencia de discapacidades definidos.

Además de los criterios mencionados también es posible cambiar las unidades de la velocidad. El programa puede mostrar las velocidades en metros por segundo o kilómetros por hora. Para cambiar las unidades solo seleccione la unidad deseada en el menú desplegable al lado del histograma.



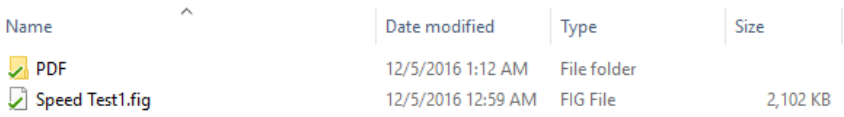


Unidades disponibles

IMPORTANTE: En cada criterio, los estados de no marcar ningún botón o marcar todos son equivalentes.

7. Guardar archivo

Para guardar el archivo pulse el botón Guardar. El programa creará un archivo dentro de la carpeta asignada a la herramienta en la carpeta del proyecto. Este archivo no debe ser manipulado por el usuario.

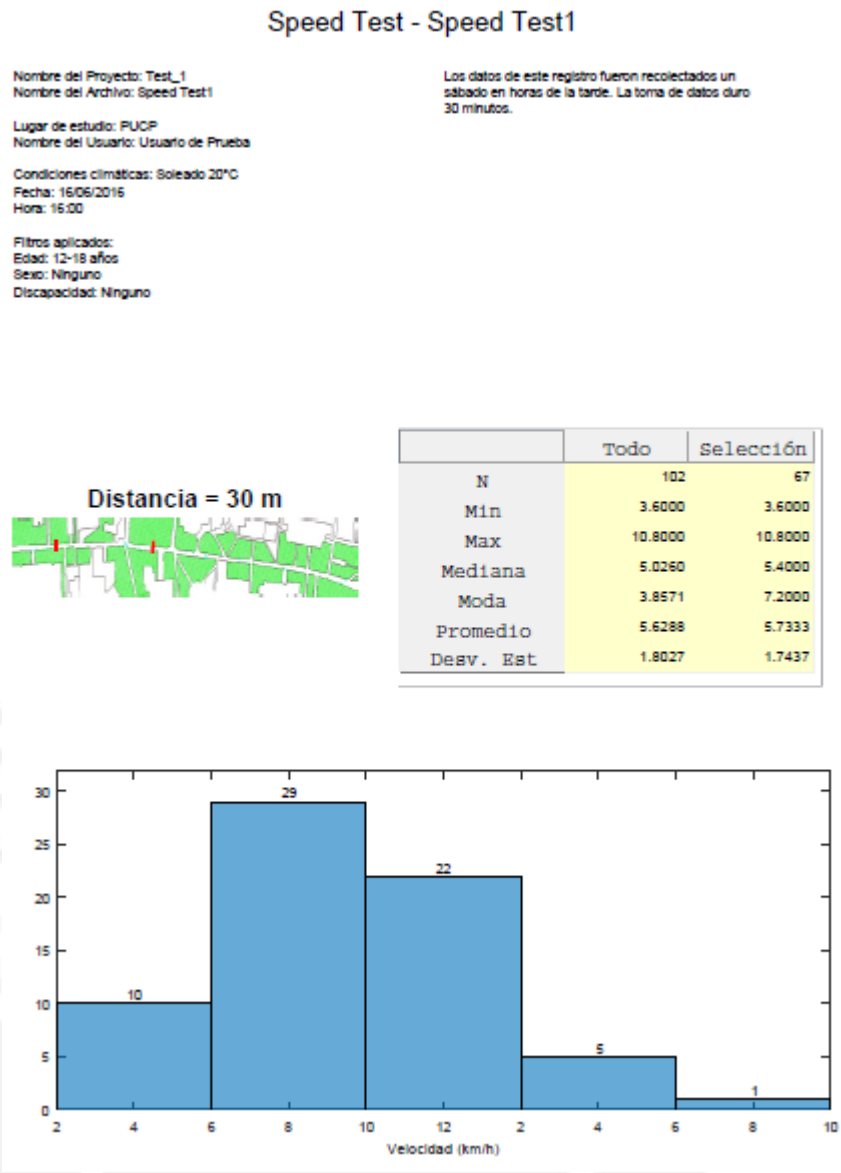


Guardar archivo.

8. Generar reportes

El programa permite generar reportes en formato PDF. Para ello basta con presionar el botón “Generar PDF”. El programa automáticamente generará un archivo en formato PDF dentro de la carpeta PDF que se muestra en la imagen anterior. La información mostrada en el reporte coincide con los criterios de visualización activos.

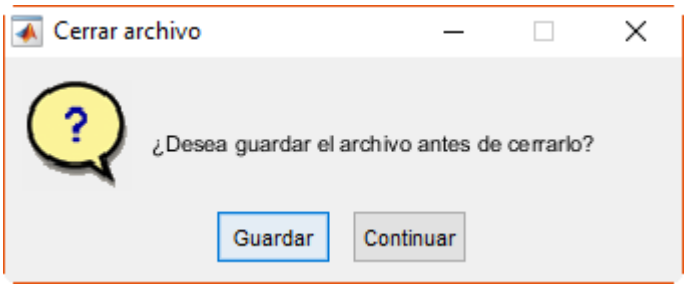
El archivo se abrirá automáticamente luego de ser creado. El programa está diseñado para no sobrescribir los reportes generados, de esta manera se pueden generar reportes con distintas configuraciones de visualización.



Reporte PDF.

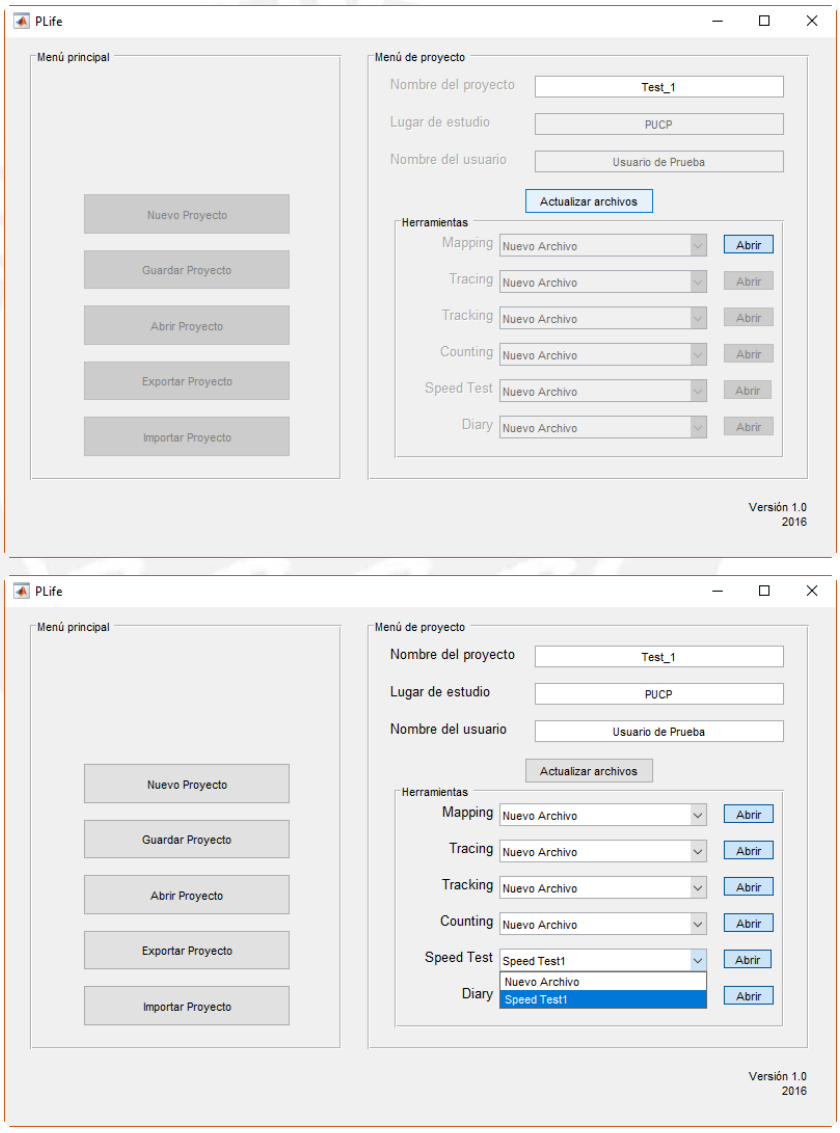
9. Cerrar herramienta

Finalmente, para regresar a la pantalla principal del programa presione el botón Cerrar. El programa consultará si desea guardar el archivo o si desea continuar sin guardar. Luego se cerrará la ventana de la herramienta y se abrirá la pantalla principal.



Cerrar herramienta.

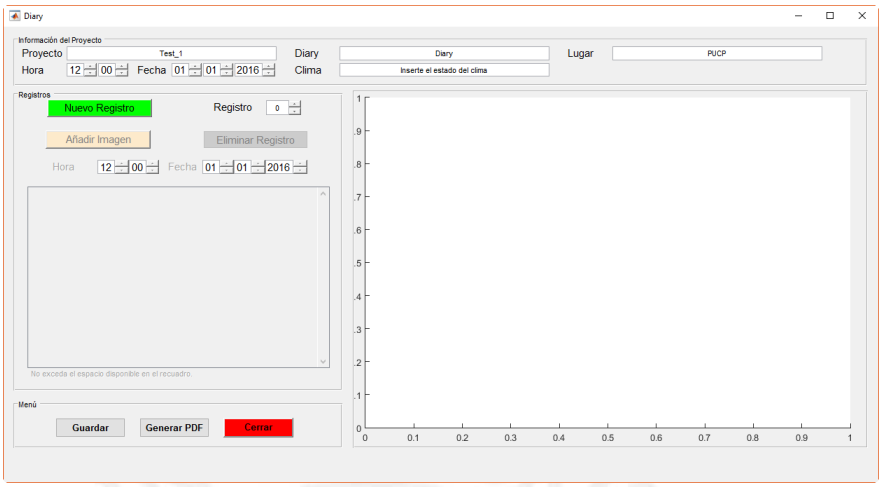
La ventana principal se abrirá con todos los elementos deshabilitados. Presione el botón Actualizar Archivos para habilitar la ventana. Si el archivo fue guardado, ahora este será visible en su respectivo menú desplegable.



Habilitar Pantalla Principal.

Capítulo 8 – Diary

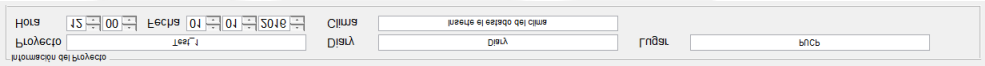
Al crear un nuevo archivo de la herramienta Diary se abre la siguiente pantalla.



Pantalla inicial Tracking.

1. Editar información del proyecto

   En la parte superior se encuentra el panel Información del Proyecto. Dentro del panel hay cuatro campos de texto. El primero muestra el nombre del proyecto; el segundo, el nombre del archivo; y el tercero, el nombre del lugar analizado. Estos recuadros no son editables y muestran la información ingresada en la pantalla principal. El cuarto recuadro permite ingresar las condiciones climáticas registradas. En el mismo panel se cuenta con botones para definir la hora y fecha del registro. Toda la información ingresada se mostrará en el reporte generado por el programa.



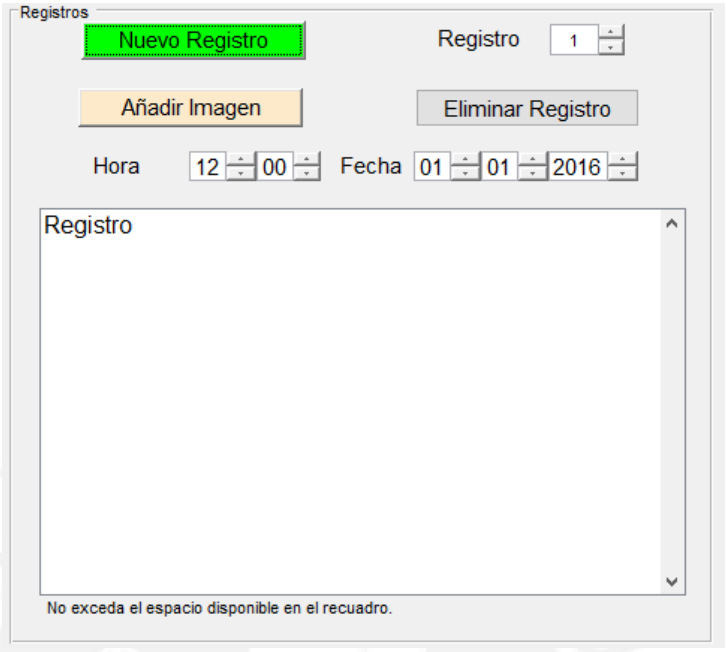
Información del Proyecto.

2. Añadir registros

   Esta herramienta funciona en base a registros. Cada registro puede contener una fotografía y una nota de texto. Además, a cada registro le corresponde una fecha y hora. Todos estos datos son ingresados desde el panel Registros.

   Para crear un registro debe presionar el botón Nuevo Registro. El contador se incrementará en uno y se habilitará la edición del registro. Puede desplazarse entre los registros existentes utilizando los botones disponibles.

Finalmente, los registros también pueden ser eliminados utilizando el botón Eliminar Registro. Tenga en cuenta que, una vez eliminado, no se puede recuperar el registro.



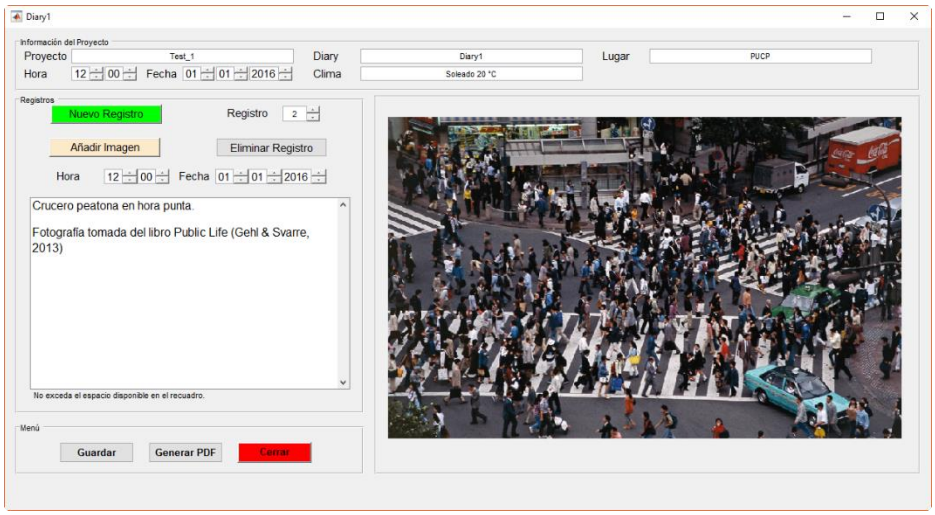
Información del Proyecto.

El texto puede añadirse libremente, pero el usuario debe tener en cuenta que si el texto ingresado supera el espacio que se muestra en pantalla entonces no se visualizará correctamente en el reporte.

3. Añadir imagen

Para importar la imagen se debe presionar el botón “Añadir Imagen”. Luego se habilita una ventana para seleccionar el archivo deseado. Luego de seleccionar la imagen esta se muestra en la pantalla de la aplicación.

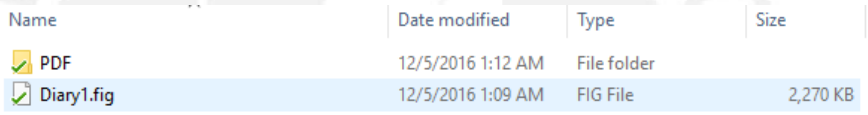
IMPORTANTE: Por defecto se muestra la carpeta Imágenes dentro del proyecto. Si el usuario selecciona un archivo que no esté dentro de esta carpeta el programa creará una copia de la imagen en la carpeta “Imágenes”. De esta manera, todas las imágenes del proyecto son almacenadas en la misma carpeta para su posterior uso desde otras aplicaciones.



Importar imagen.

4. Guardar archivo

Para guardar el archivo pulse el botón Guardar. El programa creará un archivo dentro de la carpeta asignada a la herramienta en la carpeta del proyecto. Este archivo no debe ser manipulado por el usuario.



Guardar archivo.

5. Generar reportes

El programa permite generar reportes en formato PDF. Para ello basta con presionar el botón “Generar PDF”. El programa automáticamente generará un archivo en formato PDF dentro de la carpeta PDF que se muestra en la imagen anterior. El programa creará una hoja para cada tres registros. Los archivos se abrirán automáticamente luego de ser creados.


Speed Test - Diary1

Nombre del Proyecto: Test_1
Nombre del Archivo: Diary1

Lugar de estudio: PUCP
Nombre del Usuario: Usuario de Prueba

Condiciones climáticas: Soleado 20 °C
Fecha: 01/01/2016
Hora: 12:00

Registro: 1
Fecha: 01/01/2016
Hora: 12:00

Fotografía de una calle peatonalizada.

Fotografía tomada del libro Public Life (Gehl & Svarre, 2013)

Registro: 2
Fecha: 01/01/2016
Hora: 12:00

Crucero peatona en hora punta.

Fotografía tomada del libro Public Life (Gehl & Svarre, 2013)

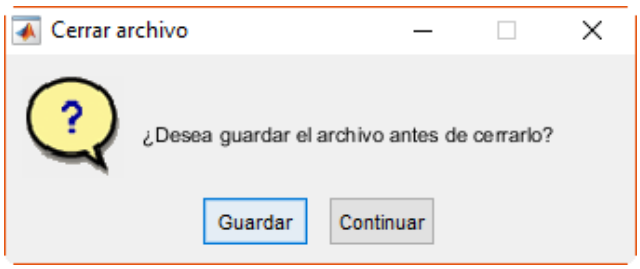
Registro: 3
Fecha: 01/01/2016
Hora: 12:00

Fotografía tomada del libro Public Life (Gehl & Svarre, 2013)

Reporte PDF.

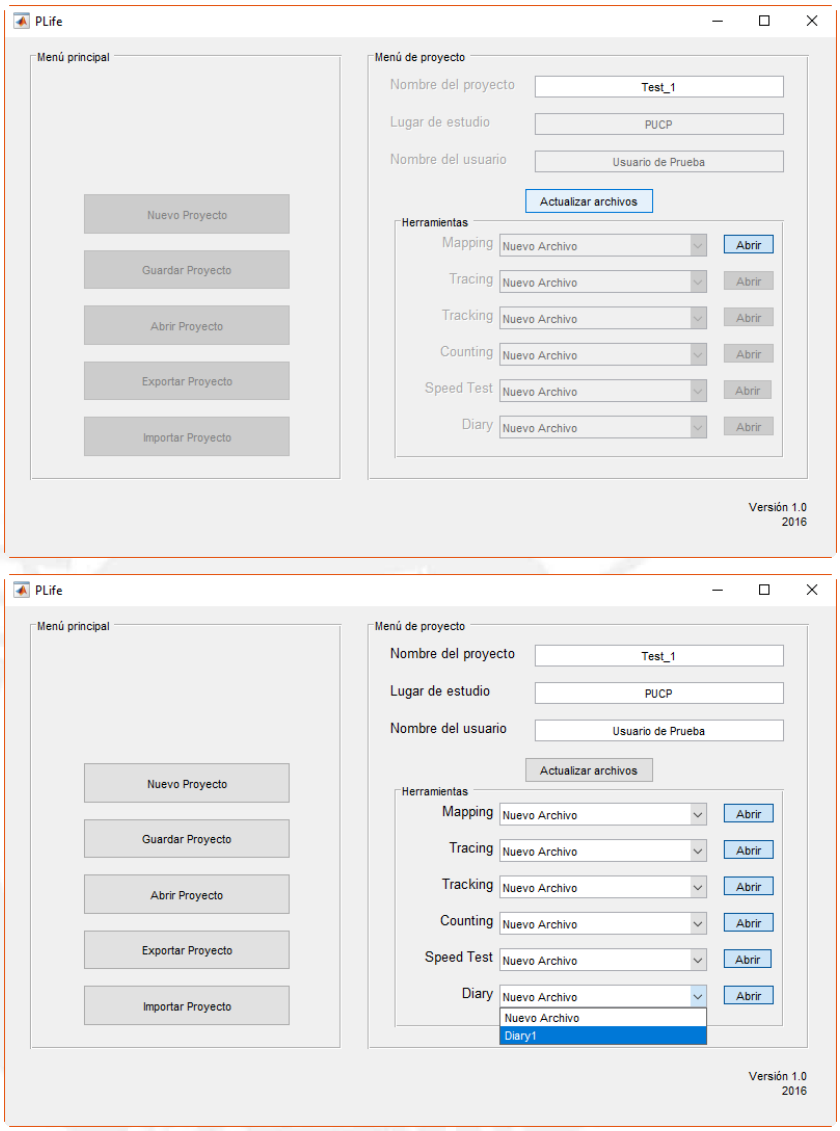
6. Cerrar herramienta

Finalmente, para regresar a la pantalla principal del programa presione el botón Cerrar. El programa consultará si desea guardar el archivo o si desea continuar sin guardar. Luego se cerrará la ventana de la herramienta y se abrirá la pantalla principal.



Cerrar herramienta.

La ventana principal se abrirá con todos los elementos deshabilitados. Presione el botón Actualizar Archivos para habilitar la ventana. Si el archivo fue guardado, ahora este será visible en su respectivo menú desplegable.



Habilitar Pantalla Principal.

ANEXO B

```
function varargout = PantallaPrincipal(varargin)
% PANTALLAPRINCIPAL MATLAB code for PantallaPrincipal.fig
%      PANTALLAPRINCIPAL, by itself, creates a new PANTALLAPRINCIPAL or raises the
existing
%      singleton*.
%
%      H = PANTALLAPRINCIPAL returns the handle to a new PANTALLAPRINCIPAL or the
handle to
%      the existing singleton*.
%
%      PANTALLAPRINCIPAL('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in PANTALLAPRINCIPAL.M with the given input arguments.
%
%      PANTALLAPRINCIPAL('Property','Value',...) creates a new PANTALLAPRINCIPAL or
raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before PantallaPrincipal_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to PantallaPrincipal_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help PantallaPrincipal

% Last Modified by GUIDE v2.5 17-May-2016 10:39:59

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @PantallaPrincipal_OpeningFcn, ...
                   'gui_OutputFcn',  @PantallaPrincipal_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before PantallaPrincipal is made visible.
function PantallaPrincipal_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to PantallaPrincipal (see VARARGIN)

warning('on','all')

handles.AppFolder = getcurrentdir;

set(findall(handles.Panel_Menu_Proyecto,'Enable','On'),'Enable','Off');

handles.PB_Guardar_Proyecto.Enable = 'Off';
handles.PB_Exportar_Proyecto.Enable = 'Off';

% Choose default command line output for PantallaPrincipal
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);
```

```
% --- Outputs from this function are returned to the command line.
function varargout = PantallaPrincipal_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --------------------------------------------------------------------
% --- Executes on button press in PB_Nuevo_Proyecto.
function PB_Nuevo_Proyecto_Callback(hObject, eventdata, handles)

    strText = sprintf(['Ingrese el nombre del nuevo proyecto.\n'...
                       'Use solo letras y/o números.']);
    strTitle = 'Nombre del proyecto';
    nombre = ingresar_nombre({strText}, strTitle,{''});
    if (isempty(nombre)) return; end

    while ( exist([handles.AppFolder '\Proyectos\' nombre],'dir') )

        choice = questdlg(sprintf(['Este proyecto ya existe.\n' ...
                                   'Si continúa el proyecto será sobrescrito.']), ...
                                   'Proyecto existente', ...
                                   'Continuar','Ingresar otro nombre','Continuar');
            switch choice
                case 'Continuar'
                    rmdir([handles.AppFolder '\Proyectos\' nombre], 's');
                case 'Ingresar otro nombre'
                    nombre = ingresar_nombre({strText}, strTitle,{''});
                    if (isempty(nombre)) return; end
            end
    end

    handles.nombre_proyecto = nombre;
    handles.lugar_estudio = 'Nuevo Lugar de Estudio';
    handles.nombre_usuario = 'Nuevo Usuario';

    handles.TE_Nombre_Proyecto.String = nombre;

    handles.path_proyecto = [handles.AppFolder '\Proyectos\' nombre];

    mkdir(handles.path_proyecto);
    mkdir([handles.path_proyecto '\Mapping\PDF']);
    mkdir([handles.path_proyecto '\Tracing\PDF']);
    mkdir([handles.path_proyecto '\Tracking\PDF']);
    mkdir([handles.path_proyecto '\Counting\PDF']);
    mkdir([handles.path_proyecto '\Speed Test\PDF']);
    mkdir([handles.path_proyecto '\Diary\PDF']);

    mkdir([handles.path_proyecto '\Imagenes']);

    set(findall(handles.Panel_Menu_Proyecto,'Enable','Off'),'Enable','On');

    handles.PB_Guardar_Proyecto.Enable = 'On';
    handles.PB_Exportar_Proyecto.Enable = 'On';

guidata(hObject, handles);

function PB_Guardar_Proyecto_Callback(hObject, eventdata, handles)

    set(findall(handles.Panel_Menu_Proyecto ,'Enable','On'),'Enable','Off');
    set(findall(handles.Panel_Menu_Principal,'Enable','On'),'Enable','Off');
    handles.PB_Actualizar.Enable = 'On';
    savefig(handles.output,[handles.path_proyecto '\Pantalla_Principal.fig']);

    set(findall(handles.Panel_Menu_Proyecto ,'Enable','Off'),'Enable','On');
    set(findall(handles.Panel_Menu_Principal,'Enable','Off'),'Enable','On');
```

```
    guidata(hObject,handles);

function PB_Abrir_Proyecto_Callback(hObject, eventdata, handles)

    proyecto_elegido = uipickfiles('FilterSpec', [handles.AppFolder
'\Proyectos\*.fig'],...
                        'Prompt', 'Seleccione el archivo Pantalla Principal.fig',...
                        'NumFiles', 1);
    if (~iscell(proyecto_elegido) && ~proyecto_elegido) return; end

    openfig(proyecto_elegido{1});


delete(handles.output);

function PB_Exportar_Proyecto_Callback(hObject, eventdata, handles)

    strText = sprintf(['Ingrese el nombre del archivo exportado.\n'...
                        'Use solo letras y/o números.']);
    strTitle = 'Nombre del archivo exportado';
    nombre = ingresar_nombre({strText}, strTitle,{handles.nombre_proyecto});
    if (isempty(nombre)) return; end

    if ~exist('Exportados','dir') mkdir('Exportados'); end

    while ( exist([handles.AppFolder '\Exportados\' nombre],'file') )
        choice = questdlg(sprintf(['Ya existe un archivo exportado con ese nombre.\n'
...
                                   'Si continúa el archivo será sobrescrito.']), ...
                                   'Archivo existente', ...
                                   'Continuar','Ingresar otro nombre','Continuar');
            switch choice
                case 'Continuar'
                    rmdir([handles.AppFolder '\Proyectos\' nombre], 's');
                case 'Ingresar otro nombre'
                    nombre = ingresar_nombre({strText},
strTitle,{handles.nombre_proyecto});
                    if (isempty(nombre)) return; end
            end
    end
    zip([handles.AppFolder '\Exportados\'
handles.nombre_proyecto],handles.path_proyecto)

function PB_Importar_Proyecto_Callback(hObject, eventdata, handles)

    proyecto_importado = uipickfiles('FilterSpec', [handles.AppFolder
'\Exportados\*.zip'],...
                         'Prompt', 'Seleccione el archivo que desea importar',...
                         'NumFiles', 1);
    if (~iscell(proyecto_importado) && ~proyecto_importado) return; end

    choice = questdlg(sprintf( ['Se importará el proyecto seleccionado\n.'...
             'Si existe un proyecto con el mismo nombre será sobrescrito.\n',...
             'Desea continuar']), 'Solicitud de confirmación','Si','No','Si');
    if (strcmp(choice, 'No') || isempty(choice) )return; end
    unzip(proyecto_importado{1},[handles.AppFolder '\Proyectos']);
    msgbox('Puede accedar al proyecto importado desde el botón Abrir Proyecto.',...
           'Proyecto importado exitosamente');

function TE_Lugar_Estudio_Callback(hObject, eventdata, handles)

    handles.lugar_estudio = hObject.String;

guidata(hObject,handles);

function TE_Nombre_Usuario_Callback(hObject, eventdata, handles)

    handles.nombre_usuario = hObject.String;

guidata(hObject, handles);
```

```
% Herramientas
% --- Executes on button press in PB_Actualizar.
function PB_Actualizar_Callback(hObject, eventdata, handles)

    handles.AppFolder = getcurrentdir;
    handles.path_proyecto = [handles.AppFolder '\Proyectos\'
handles.nombre_proyecto];

    FileList = getAllFiles([handles.path_proyecto '\Counting'], '*.fig', 0, 0);
    for i = 1:length(FileList)
        FileList{i} = FileList{i}(1:end-4);
    end
    handles.PM_Counting.String = [{'Nuevo Archivo'}; FileList];

    FileList = getAllFiles([handles.path_proyecto '\Tracing'], '*.fig', 0, 0);
    for i = 1:length(FileList)
        FileList{i} = FileList{i}(1:end-4);
    end
    handles.PM_Tracing.String = [{'Nuevo Archivo'}; FileList];

    FileList = getAllFiles([handles.path_proyecto '\Tracking'], '*.fig', 0, 0);
    for i = 1:length(FileList)
        FileList{i} = FileList{i}(1:end-4);
    end
    handles.PM_Tracking.String = [{'Nuevo Archivo'}; FileList];

    FileList = getAllFiles([handles.path_proyecto '\Speed Test'], '*.fig', 0, 0);
    for i = 1:length(FileList)
        FileList{i} = FileList{i}(1:end-4);
    end
    handles.PM_Speed_Test.String = [{'Nuevo Archivo'}; FileList];

    FileList = getAllFiles([handles.path_proyecto '\Mapping'], '*.fig', 0, 0);
    for i = 1:length(FileList)
        FileList{i} = FileList{i}(1:end-4);
    end
    handles.PM_Mapping.String = [{'Nuevo Archivo'}; FileList];

    FileList = getAllFiles([handles.path_proyecto '\Diary'], '*.fig', 0, 0);
    for i = 1:length(FileList)
        FileList{i} = FileList{i}(1:end-4);
    end
    handles.PM_Diary.String = [{'Nuevo Archivo'}; FileList];

    set(findall(handles.Panel_Menu_Proyecto ,'Enable','Off'),'Enable','On');
    set(findall(handles.Panel_Menu_Principal,'Enable','Off'),'Enable','On');
guidata(hObject, handles);

% --- Executes on button press in PB_Mapping.
function PB_Mapping_Callback(hObject, eventdata, handles)
    PB_Guardar_Proyecto_Callback(handles.PB_Guardar_Proyecto, eventdata, handles)
    opcion_elegida = handles.PM_Mapping.Value;

    if(opcion_elegida == 1)
        strText = sprintf(['Ingrese el nombre del nuevo archivo.\n'...
                  'Use solo letras y/o números.']);
        strTitle = 'Nuevo nombre de archivo' ;
        nombre = ingresar_nombre({strText}, strTitle,{''});
        if (isempty(nombre)) return; end

        while (exist([handles.path_proyecto '\Mapping\' nombre],'file'))
            choice = questdlg(sprintf(['Este arvhivo ya existe.\n' ...
                      'Si continúa el archivo será sobrescrito.']), ...
                      'Archivo existente', ...
                      'Continuar','Ingresar otro nombre','Continuar');
                switch choice
                    case 'Continuar'
                        delete([handles.path_proyecto '\Mapping\' nombre]);
                    case 'Ingresar otro nombre'
                        nombre = ingresar_nombre({strText}, strTitle,{''});
                        if (isempty(nombre)) return; end
                end
        end
```

```
            PB_Guardar_Proyecto_Callback(handles.PB_Guardar_Proyecto, eventdata, handles)
            Mapping({handles.AppFolder, handles.path_proyecto, ...
                     handles.nombre_proyecto, nombre,...
                     handles.lugar_estudio, handles.nombre_usuario});

        else

            PB_Guardar_Proyecto_Callback(handles.PB_Guardar_Proyecto, eventdata, handles)
            h = openfig([handles.path_proyecto '\Mapping\' ...
                     handles.PM_Mapping.String{opcion_elegida} '.fig']);

            new_handles = guidata (h);
            new_handles.AppFolder = handles.AppFolder;
            new_handles.path_proyecto = handles.path_proyecto;
            guidata(h,new_handles);
        end

    delete(handles.output);

    % --- Executes on button press in PB_Tracing.
    function PB_Tracing_Callback(hObject, eventdata, handles)

        PB_Guardar_Proyecto_Callback(handles.PB_Guardar_Proyecto, eventdata, handles)
        opcion_elegida = handles.PM_Tracing.Value;

        if(opcion_elegida == 1)
            strText = sprintf(['Ingrese el nombre del nuevo archivo.\n'...
                      'Use solo letras y/o números.']);
            strTitle = 'Nuevo nombre de archivo' ;
            nombre = ingresar_nombre({strText}, strTitle,{''});
            if (isempty(nombre)) return; end

            while (exist([handles.path_proyecto '\Tracing\' nombre],'file'))
                choice = questdlg(sprintf(['Este arvhivo ya existe.\n' ...
                         'Si continúa el archivo será sobrescrito.']), ...
                         'Archivo existente', ...
                         'Continuar','Ingresar otro nombre','Continuar');
                    switch choice
                        case 'Continuar'
                            delete([handles.path_proyecto '\Tracing\' nombre]);
                        case 'Ingresar otro nombre'
                            nombre = ingresar_nombre({strText}, strTitle,{''});
                            if (isempty(nombre)) return; end
                    end
            end

            PB_Guardar_Proyecto_Callback(handles.PB_Guardar_Proyecto, eventdata, handles)
            Tracing({handles.AppFolder, handles.path_proyecto, ...
                     handles.nombre_proyecto, nombre,...
                     handles.lugar_estudio, handles.nombre_usuario});

        else

            PB_Guardar_Proyecto_Callback(handles.PB_Guardar_Proyecto, eventdata, handles)
            h = openfig([handles.path_proyecto '\Tracing\' ...
                     handles.PM_Tracing.String{opcion_elegida} '.fig']);
            new_handles = guidata (h);
            new_handles.AppFolder = handles.AppFolder;
            new_handles.path_proyecto = handles.path_proyecto;
            guidata(h,new_handles);

        end

    delete(handles.output);

    % --- Executes on button press in PB_Tracking.
    function PB_Tracking_Callback(hObject, eventdata, handles)
    % hObject    handle to PB_Tracking (see GCBO)
        PB_Guardar_Proyecto_Callback(handles.PB_Guardar_Proyecto, eventdata, handles)
        opcion_elegida = handles.PM_Tracking.Value;

        if(opcion_elegida == 1)
            strText = sprintf(['Ingrese el nombre del nuevo archivo.\n'...
```

```
                'Use solo letras y/o números.']);
        strTitle = 'Nuevo nombre de archivo' ;
        nombre = ingresar_nombre({strText}, strTitle,{''});
        if (isempty(nombre)) return; end

        while (exist([handles.path_proyecto '\Tracking\' nombre],'file'))
            choice = questdlg(sprintf(['Este arvhivo ya existe.\n' ...
                     'Si continúa el archivo será sobrescrito.']), ...
                     'Archivo existente', ...
                     'Continuar','Ingresar otro nombre','Continuar');
                switch choice
                    case 'Continuar'
                        delete([handles.path_proyecto '\Tracking\' nombre]);
                    case 'Ingresar otro nombre'
                        nombre = ingresar_nombre({strText}, strTitle,{''});
                        if (isempty(nombre)) return; end
                end
        end

        PB_Guardar_Proyecto_Callback(handles.PB_Guardar_Proyecto, eventdata, handles)
        Tracking({handles.AppFolder, handles.path_proyecto, ...
                 handles.nombre_proyecto, nombre,...
                 handles.lugar_estudio, handles.nombre_usuario});

    else

        PB_Guardar_Proyecto_Callback(handles.PB_Guardar_Proyecto, eventdata, handles)
        h = openfig([handles.path_proyecto '\Tracking\' ...
                handles.PM_Tracking.String{opcion_elegida} '.fig']);

        new_handles = guidata (h);
        new_handles.AppFolder = handles.AppFolder;
        new_handles.path_proyecto = handles.path_proyecto;
        guidata(h,new_handles);

    end

delete(handles.output);

% --- Executes on button press in PB_Counting.
function PB_Counting_Callback(hObject, eventdata, handles)

    PB_Guardar_Proyecto_Callback(handles.PB_Guardar_Proyecto, eventdata, handles)
    opcion_elegida = handles.PM_Counting.Value;

    if(opcion_elegida == 1)
        strText = sprintf(['Ingrese el nombre del nuevo archivo.\n'...
                  'Use solo letras y/o números.']);
        strTitle = 'Nuevo nombre de archivo' ;
        nombre = ingresar_nombre({strText}, strTitle,{''});
        if (isempty(nombre)) return; end

        while (exist([handles.path_proyecto '\Counting\' nombre],'file'))
            choice = questdlg(sprintf(['Este arvhivo ya existe.\n' ...
                     'Si continúa el archivo será sobrescrito.']), ...
                     'Archivo existente', ...
                     'Continuar','Ingresar otro nombre','Continuar');
                switch choice
                    case 'Continuar'
                        delete([handles.path_proyecto '\Counting\' nombre]);
                    case 'Ingresar otro nombre'
                        nombre = ingresar_nombre({strText}, strTitle,{''});
                        if (isempty(nombre)) return; end
                end
        end

        PB_Guardar_Proyecto_Callback(handles.PB_Guardar_Proyecto, eventdata, handles)
        Counting({handles.AppFolder, handles.path_proyecto, ...
                  handles.nombre_proyecto, nombre,...
                  handles.lugar_estudio, handles.nombre_usuario});

    else
```

```
        PB_Guardar_Proyecto_Callback(handles.PB_Guardar_Proyecto, eventdata, handles)
        h = openfig([handles.path_proyecto '\Counting\' ...
                 handles.PM_Counting.String{opcion_elegida} '.fig']);

        new_handles = guidata (h);
        new_handles.AppFolder = handles.AppFolder;
        new_handles.path_proyecto = handles.path_proyecto;
        guidata(h,new_handles);
    end

delete(handles.output);

% --- Executes on button press in PB_Speed_Test.
function PB_Speed_Test_Callback(hObject, eventdata, handles)

    PB_Guardar_Proyecto_Callback(handles.PB_Guardar_Proyecto, eventdata, handles)
    opcion_elegida = handles.PM_Speed_Test.Value;

    if(opcion_elegida == 1)
        strText = sprintf(['Ingrese el nombre del nuevo archivo.\n'...
                  'Use solo letras y/o números.']);
        strTitle = 'Nuevo nombre de archivo' ;
        nombre = ingresar_nombre({strText}, strTitle,{''});
        if (isempty(nombre)) return; end

        while (exist([handles.path_proyecto '\Speed Test\' nombre],'file'))
            choice = questdlg(sprintf(['Este arvhivo ya existe.\n' ...
                     'Si continúa el archivo será sobrescrito.']), ...
                     'Archivo existente', ...
                     'Continuar','Ingresar otro nombre','Continuar');
                 switch choice
                     case 'Continuar'
                         delete([handles.path_proyecto '\Speed Test\' nombre]);
                     case 'Ingresar otro nombre'
                         nombre = ingresar_nombre({strText}, strTitle,{''});
                         if (isempty(nombre)) return; end
                 end
        end

        PB_Guardar_Proyecto_Callback(handles.PB_Guardar_Proyecto, eventdata, handles)
        Speed_Test({handles.AppFolder, handles.path_proyecto, ...
                    handles.nombre_proyecto, nombre,...
                    handles.lugar_estudio, handles.nombre_usuario});

    else

        PB_Guardar_Proyecto_Callback(handles.PB_Guardar_Proyecto, eventdata, handles)
        h = openfig([handles.path_proyecto '\Speed Test\' ...
                 handles.PM_Speed_Test.String{opcion_elegida} '.fig']);

        new_handles = guidata (h);
        new_handles.AppFolder = handles.AppFolder;
        new_handles.path_proyecto = handles.path_proyecto;
        guidata(h,new_handles);
    end

delete(handles.output);

% --- Executes on button press in PB_Diary.
function PB_Diary_Callback(hObject, eventdata, handles)
    PB_Guardar_Proyecto_Callback(handles.PB_Guardar_Proyecto, eventdata, handles)
    opcion_elegida = handles.PM_Diary.Value;

    if(opcion_elegida == 1)
        strText = sprintf(['Ingrese el nombre del nuevo archivo.\n'...
                  'Use solo letras y/o números.']);
        strTitle = 'Nuevo nombre de archivo' ;
        nombre = ingresar_nombre({strText}, strTitle,{''});
        if (isempty(nombre)) return; end
        while (exist([handles.path_proyecto '\Diary\' nombre],'file'))
            choice = questdlg(sprintf(['Este arvhivo ya existe.\n'
                     'Si continúa el archivo será sobrescrito.']),
```

```
                    'Archivo existente', ...
                    'Continuar','Ingresar otro nombre','Continuar');
                switch choice
                    case 'Continuar'
                        delete([handles.path_proyecto '\Diary\' nombre]);
                    case 'Ingresar otro nombre'
                        nombre = ingresar_nombre({strText}, strTitle,{''});
                        if (isempty(nombre)) return; end
                end
            end

            PB_Guardar_Proyecto_Callback(handles.PB_Guardar_Proyecto, eventdata, handles)
            Diary({handles.AppFolder, handles.path_proyecto, ...
                     handles.nombre_proyecto, nombre,...
                     handles.lugar_estudio, handles.nombre_usuario});

        else

            PB_Guardar_Proyecto_Callback(handles.PB_Guardar_Proyecto, eventdata, handles)
            h = openfig([handles.path_proyecto '\Diary\' ...
                     handles.PM_Diary.String{opcion_elegida} '.fig']);

            new_handles = guidata (h);
            new_handles.AppFolder = handles.AppFolder;
            new_handles.path_proyecto = handles.path_proyecto;
            guidata(h,new_handles);
        end

    delete(handles.output);
```

```
function varargout = Tracing(varargin)
% TRACING MATLAB code for Tracing.fig
%      TRACING, by itself, creates a new TRACING or raises the existing
%      singleton*.
%
%      H = TRACING returns the handle to a new TRACING or the handle to
%      the existing singleton*.
%
%      TRACING('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in TRACING.M with the given input arguments.
%
%      TRACING('Property','Value',...) creates a new TRACING or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before Tracing_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to Tracing_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help Tracing

% Last Modified by GUIDE v2.5 04-Dec-2016 02:00:30

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Tracing_OpeningFcn, ...
                   'gui_OutputFcn',  @Tracing_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

% --- Executes just before Tracing is made visible.
function Tracing_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to Tracing (see VARARGIN)

warning('off','all')
set(findall(handles.Tracing,'Units','Pixels'),'Units','Normalized');

handles.AppFolder = varargin{1}{1};
handles.path_proyecto = varargin{1}{2};
handles.nombre_proyecto = varargin{1}{3};
handles.nombre_archivo = varargin{1}{4};
handles.lugar_estudio = varargin{1}{5};
handles.nombre_usuario = varargin{1}{6};
handles.Tracing.Name = handles.nombre_archivo;

handles.TE_Proyecto.String = handles.nombre_proyecto;
handles.TE_Archivo.String = handles.nombre_archivo;
handles.TE_Lugar.String = handles.lugar_estudio;
handles.Clima = '';
handles.Notas = 'Ingrese sus notas';

handles.Visu.Sexo = 0;
handles.Visu.Edad = 0;
```

```
handles.Visu.Disc = 0;

handles.Info_Trazos = cell(1);

set(findall(handles.Panel_Trazos,'Enable','On'),'Enable','Off');
set(findall(handles.Panel_Menu_Trazos,'Enable','On'),'Enable','Off');
set(findall(handles.Panel_Menu,'Enable','On'),'Enable','Off');

handles.PB_Cerrar.Enable = 'On';

% Choose default command line output for Tracing
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes Tracing wait for user response (see UIRESUME)
% uiwait(handles.Tracing);

% --- Outputs from this function are returned to the command line.
function varargout = Tracing_OutputFcn(hObject, eventdata, handles)

% Get default command line output from handles structure
varargout{1} = handles.output;

%--------------------------------------------------------------------------
% Panel Informacion
% --- Executes on slider movement.
function SL_Hora_Callback(hObject, eventdata, handles)

    valor = round(hObject.Value);
    handles.TE_Hora.String = num2str(valor,'%02u');

guidata(hObject, handles);

% --- Executes on slider movement.
function SL_Minuto_Callback(hObject, eventdata, handles)

    valor = round(hObject.Value);
    handles.TE_Minuto.String = num2str(valor,'%02u');

guidata(hObject, handles);

% --- Executes on slider movement.
function SL_Dia_Callback(hObject, eventdata, handles)

    valor = round(hObject.Value);
    handles.TE_Dia.String = num2str(valor,'%02u');

guidata(hObject, handles);

% --- Executes on slider movement.
function SL_Mes_Callback(hObject, eventdata, handles)

    valor = round(hObject.Value);
    handles.TE_Mes.String = num2str(valor,'%02u');

guidata(hObject, handles);

% --- Executes on slider movement.
function SL_Anio_Callback(hObject, eventdata, handles)

    valor = round(hObject.Value);
    handles.TE_Anio.String = num2str(valor,'%04u');

guidata(hObject, handles);

function TE_Clima_Callback(hObject, eventdata, handles)

    handles.Clima = hObject.String;
```

```
guidata(hObject,handles)

% --- Executes on button press in PB_Notas.
function PB_Notas_Callback(hObject, eventdata, handles)

    answer = inputdlg({'Ingrese las notas - No exceda el espacio en la ventana'},...
                      'Notas - Tracing', [12 50], {handles.Notas});
    if (isempty(answer)) return; end

    handles.Notas = answer{:};
guidata(hObject,handles);

% --- Executes on button press in PB_Importar_Imagen.
function PB_Importar_Imagen_Callback(hObject, eventdata, handles)

    image_path = uipickfiles('FilterSpec', [handles.AppFolder...
                 '\Proyectos\' handles.nombre_proyecto '\Imagenes\*.jpg'],...
                 'Prompt', 'Seleccione un archivo JPG', 'NumFiles', 1);

    if (~iscell(image_path) && ~image_path) return; end

    cla(handles.Axes)

    if (isempty(strfind(image_path{1},[handles.AppFolder...
        '\Proyectos\' handles.nombre_proyecto '\Imagenes\'])))

        copyfile(image_path{1},[handles.AppFolder '\Proyectos\'...
                 handles.nombre_proyecto '\Imagenes\Tracing_'...
                 handles.nombre_archivo '.jpg']);
        image_path{1} = [handles.AppFolder '\Proyectos\'...
                 handles.nombre_proyecto '\Imagenes\Tracing_'...
                 handles.nombre_archivo '.jpg'];
    end
    handles.data_imagen = imread( image_path{1} );
    axes(handles.Axes);
    handles.Imagen = imshow (handles.data_imagen); hold on;
    axis image;

    set(findall(handles.Panel_Trazos,'Enable','Off'),'Enable','On');
    set(findall(handles.Panel_Menu,'Enable','Off'),'Enable','On');
    hObject.Enable = 'Off';


guidata(hObject,handles);

% Panel Trazos
% --- Executes on button press in PB_Agregar_Trazo.
function PB_Agregar_Trazo_Callback(hObject, eventdata, handles)

    Lista_Trazos = handles.PM_Trazos.String;
    if (~iscell(Lista_Trazos)) Lista_Trazos = {'Nuevo Trazo'}; end
    pos = 1;
    while(1)
        if (~any(strcmp(Lista_Trazos,['Trazo ' num2str(pos)]))) break; end
        pos = pos + 1;
    end

    strText = sprintf(['Ingrese el nombre del nuevo trazo.\n'...
                       'Use solo letras y/o números.']);
    strTitle = 'Nombre de trazo';
    nombre = ingresar_nombre({strText}, strTitle,{['Trazo ' num2str(pos)]});

    if (isempty(nombre)) return; end

    while ( any(strcmp(Lista_Trazos,nombre) ) )
        choice = questdlg(sprintf(['Este nombre de trazo ya existe.\n' ...
                 'Si continúa el trazo será sobrescrito.']), ...
                 'Nombre de trazo existente', ...
                 'Continuar','Ingresar otro nombre','Continuar');
            switch choice
                case 'Continuar'
```

```
                    pos = find( strcmp(Lista_Trazos,nombre) == 1);
                    handles.PM_Trazos.String(pos) = [];
                    delete(handles.Info_Trazos{pos}.Plot);
                    handles.Info_Trazos(pos) = [];
                    Lista_Trazos = handles.PM_Trazos.String;
                case 'Ingresar otro nombre'
                    nombre = ingresar_nombre(nombre);
                    if (isempty(nombre)) return; end
            end
    end
    Lista_Trazos{end+1} = nombre;
    handles.PM_Trazos.String = Lista_Trazos;
    handles.PM_Trazos.Value = length(Lista_Trazos);
    handles.Trazo_Actual = length(Lista_Trazos);
    handles.Info_Trazos{end+1} = struct('Nombre_Trazo', nombre,...
                                'Color', [0 0 0], 'Sexo', 1,...
                                'Edad', 1, 'Disc', 1,...
                                'X', [], 'Y', [], 'Plot' , []);
    guidata(hObject,handles);
    PB_Modificar_Trazo_Callback(handles.PB_Modificar_Trazo, eventdata, handles);
    handles = guidata(hObject);
guidata(hObject,handles);

% --- Executes on button press in PB_Duplicar_Trazo.
function PB_Duplicar_Trazo_Callback(hObject, eventdata, handles)

    pos = handles.PM_Trazos.Value;
    Lista_Trazos = handles.PM_Trazos.String;
    if (pos == 1)
        msgbox('Seleccione un trazo válido', 'Error','error');
        return;
    end

    strText = sprintf(['Ingrese el nombre del nuevo trazo.\n'...
                       'Use solo letras y/o números.']);
    strTitle = 'Nombre de trazo';
    nombre = ingresar_nombre({strText}, strTitle,{[Lista_Trazos{pos} ' 2']});

    if (isempty(nombre)) return; end

    while ( any(strcmp(Lista_Trazos,nombre) ) )
        choice = questdlg(sprintf(['El nombre ingresado ya existe.\n' ...
                 'Si continúa el trazo será sobrescrito.'...
                 'Elija un nombre distinto al trazo que desea duplicar']),...
                 'Nombre de trazo existente', ...
                 'Continuar','Ingresar otro nombre','Continuar');
            if (find( strcmp(Lista_Trazos,nombre) == 1) == pos)
                choice = 'Ingresar otro nombre';
            end
            switch choice
                case 'Continuar'

                    pos_borrar = find( strcmp(Lista_Trazos,nombre) == 1);
                    handles.PM_Trazos.String(pos_borrar) = [];
                    delete(handles.Info_Trazos{pos_borrar}.Plot);
                    handles.Info_Trazos(pos_borrar) = [];
                    Lista_Trazos = handles.PM_Trazos.String;
                case 'Ingresar otro nombre'
                    nombre = ingresar_nombre(nombre);
                    if (isempty(nombre)) return; end
            end
    end

    Lista_Trazos{end+1} = nombre;
    handles.PM_Trazos.String = Lista_Trazos;
    handles.PM_Trazos.Value = length(Lista_Trazos);

    handles.Info_Trazos{end+1} = handles.Info_Trazos{pos};
    handles.Info_Trazos{end}.Nombre_Trazo = nombre;

    handles.Trazo_Actual = length(Lista_Trazos);

    handles.Info_Trazos{handles.Trazo_Actual}.Plot = ...
        plot(handles.Info_Trazos{handles.Trazo_Actual}.X ,...
```

```
                handles.Info_Trazos{handles.Trazo_Actual}.Y ,...
                'Color',handles.PM_Color.BackgroundColor, 'LineWidth', 2);

    guidata(hObject,handles);
    PB_Modificar_Trazo_Callback(handles.PB_Modificar_Trazo, eventdata, handles);
    handles = guidata(hObject);

guidata(hObject,handles);

% --- Executes on button press in PB_Modificar_Trazo.
function PB_Modificar_Trazo_Callback(hObject, eventdata, handles)

    handles.Trazo_Actual = handles.PM_Trazos.Value;

    if (handles.Trazo_Actual == 1)
        msgbox('Seleccione un Trazo válido para modificar', 'Error','error');
        return;
    end

    set(findall(handles.Panel_Menu_Trazos,'Enable','Off'),'Enable','On');
    set(findall(handles.Panel_Trazos,'Enable','On'),'Enable','Off');
    set(findall(handles.Panel_Menu,'Enable','On'),'Enable','Off');

    Trazo = handles.Info_Trazos{handles.Trazo_Actual};

    handles.TE_Nombre_Trazo.String = Trazo.Nombre_Trazo;
    handles.PM_Color.BackgroundColor = Trazo.Color;
    guidata(hObject,handles);
    actualizar_panel_color(Trazo.Color,handles,0);
    handles = guidata(hObject);
    handles.PM_Sexo.Value = Trazo.Sexo;
    handles.PM_Edad.Value = Trazo.Edad;
    handles.PM_Discapacidad.Value = Trazo.Disc;

    for i = 2:length(handles.Info_Trazos)
        if ~isempty(handles.Info_Trazos{i}.Plot)
            handles.Info_Trazos{i}.Plot.Visible = 'Off';
        end
    end

    Trazo.Plot.Visible = 'On';

guidata(hObject,handles);

function PB_Eliminar_Trazo_Callback(hObject, eventdata, handles)

    handles.Trazo_Actual = handles.PM_Trazos.Value;
    handles.PM_Trazos.Value = 1;

    if (handles.Trazo_Actual == 1)
        msgbox('Seleccione un Trazo válido para eliminar', 'Error','error');
        return;
    end

    choice = questdlg(sprintf(['¿Desea eliminar el Trazo seleccionado?.\n' ...
                               'Una vez eliminado no podrá recuperarlo.']), ...
                               'Eliminar Trazo', ...
                               'Eliminar','Cancelar','Cancelar');
    switch choice
        case 'Eliminar'
            handles.PM_Trazos.String(handles.Trazo_Actual) = [];
            delete(handles.Info_Trazos{handles.Trazo_Actual}.Plot);
            handles.Info_Trazos(handles.Trazo_Actual) = [];
        case 'Cancelar'
            return;
    end
    handles.Trazo_Actual = 1;

guidata(hObject,handles);

%Panel Menu Trazo
% --- Executes on button press in PB_Iniciar_Trazo.
```

```
function PB_Iniciar_Trazo_Callback(hObject, eventdata, handles)

    if(~isempty(handles.Info_Trazos{handles.Trazo_Actual}.Plot))

        delete(handles.Info_Trazos{handles.Trazo_Actual}.Plot);

        handles.Info_Trazos{handles.Trazo_Actual}.Plot = [];
        handles.Info_Trazos{handles.Trazo_Actual}.X = [];
        handles.Info_Trazos{handles.Trazo_Actual}.Y = [];
    end

    set(findall(handles.Panel_Menu_Trazos,'Enable','On'),'Enable','Off');

    set(handles.Axes,'ButtonDownFcn',{@startDragFcn,handles.Tracing});
    set(handles.Imagen,'ButtonDownFcn',{@startDragFcn,handles.Tracing});

guidata(hObject, handles);

function startDragFcn(varargin)
    handles = guidata(varargin{3});
    set( handles.Tracing, 'WindowButtonMotionFcn', {@draggingFcn,handles.Tracing} );
    set( handles.Tracing, 'WindowButtonUpFcn', {@stopDragFcn,handles.Tracing});
    pt = get(handles.Axes, 'CurrentPoint');
    x = pt(1,1);
    y = pt(1,2);
    axes(handles.Axes);
    handles.Info_Trazos{handles.Trazo_Actual}.X = x;
    handles.Info_Trazos{handles.Trazo_Actual}.Y = y;

    handles.Info_Trazos{handles.Trazo_Actual}.Plot = ...
        plot(handles.Info_Trazos{handles.Trazo_Actual}.X ,...
             handles.Info_Trazos{handles.Trazo_Actual}.Y ,...
             'Color',handles.PM_Color.BackgroundColor, 'LineWidth', 1);

guidata(varargin{3}, handles);

function draggingFcn(varargin)
    handles = guidata(varargin{3});
    pt = get(handles.Axes, 'CurrentPoint');

    x = pt(1,1);
    y = pt(1,2);

    handles.Info_Trazos{handles.Trazo_Actual}.X =
[handles.Info_Trazos{handles.Trazo_Actual}.X x];
    handles.Info_Trazos{handles.Trazo_Actual}.Y =
[handles.Info_Trazos{handles.Trazo_Actual}.Y y];
    handles.Info_Trazos{handles.Trazo_Actual}.Plot.XData =
handles.Info_Trazos{handles.Trazo_Actual}.X;
    handles.Info_Trazos{handles.Trazo_Actual}.Plot.YData =
handles.Info_Trazos{handles.Trazo_Actual}.Y;

guidata(varargin{3}, handles);

function stopDragFcn(varargin)
    handles = guidata(varargin{3});
    set(handles.Tracing, 'WindowButtonMotionFcn', '');
    set(handles.Tracing, 'WindowButtonUpFcn', '');
    set(handles.Axes,'ButtonDownFcn','');
    set(handles.Imagen,'ButtonDownFcn','');

    set(findall(handles.Panel_Menu_Trazos,'Enable','Off'),'Enable','On');

guidata(varargin{3}, handles);

% --- Executes on button press in PB_Guardar_Trazo.
function PB_Guardar_Trazo_Callback(hObject, eventdata, handles)

    if (isempty(handles.Info_Trazos{handles.Trazo_Actual}.Plot))
        choice = questdlg(sprintf(['No se ha dibujado el trazo.\n' ...
```

```
                                'Si continúa el trazo ingresado no será guardado.']),
...
                                'Trazo sin dibujar', ...
                                'Continuar','Cancelar','Cancelar');
        if (strcmp(choice,'Continuar'))
            handles.PM_Trazos.String(handles.Trazo_Actual) = [];
            handles.Info_Trazos(handles.Trazo_Actual) = [];
            handles.PM_Trazos.Value = 1;
        end
    else
        handles.Info_Trazos{handles.Trazo_Actual}.Color =
handles.PM_Color.BackgroundColor;
        handles.Info_Trazos{handles.Trazo_Actual}.Sexo = handles.PM_Sexo.Value;
        handles.Info_Trazos{handles.Trazo_Actual}.Edad = handles.PM_Edad.Value;
        handles.Info_Trazos{handles.Trazo_Actual}.Disc =
handles.PM_Discapacidad.Value;
    end
    handles.TE_Nombre_Trazo.String = '';
    handles.PM_Color.BackgroundColor = [1 1 1];

    handles.PM_Trazos.Value = 1;
    handles.Trazo_Actual = 1;

    actualizar_panel_color([1 1 1],handles,0);
    handles = guidata(hObject);
    handles.PM_Sexo.Value = 1;
    handles.PM_Edad.Value = 1;
    handles.PM_Discapacidad.Value = 1;

    guidata(hObject, handles);
    set(findall(handles.Panel_Menu_Trazos,'Enable','On'),'Enable','Off');
    set(findall(handles.Panel_Trazos,'Enable','Off'),'Enable','On');
    set(findall(handles.Panel_Menu,'Enable','Off'),'Enable','On');

    Actualizar_visualizacion(hObject,handles.Axes);
guidata(hObject, handles);

% --- Executes on selection change in PM_Color.
function PM_Color_Callback(hObject, eventdata, handles)
% hObject    handle to PM_Color (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

    colores = {[], [1 1 0],[1 0 1],[0 1 1],[1 0 0],[0 1 0],[0 0 1],[1 1 1],[0 0 0]};

    if (hObject.Value ~= 1)
        hObject.BackgroundColor = colores{hObject.Value};
        hObject.Value = 1;
        col = hObject.BackgroundColor;
        actualizar_panel_color(col,handles,1);
        handles = guidata(hObject);
    end
guidata(hObject,handles);

% --- Executes on slider movement.
function SL_ColorR_Callback(hObject, eventdata, handles)
    col = handles.PM_Color.BackgroundColor;
    col(1) = round(hObject.Value)/255;
    handles.Info_Trazos{handles.Trazo_Actual}.Color = col;
    handles.PM_Color.BackgroundColor = col;
    actualizar_panel_color(col,handles,1);
    handles = guidata(hObject);
guidata(hObject,handles)

% --- Executes on slider movement.
function SL_ColorG_Callback(hObject, eventdata, handles)

    col = handles.PM_Color.BackgroundColor;
    col(2) = round(hObject.Value)/255;
    handles.Info_Trazos{handles.Trazo_Actual}.Color = col;
    handles.PM_Color.BackgroundColor = col;
    actualizar_panel_color(col,handles,1);
    handles = guidata(hObject);
guidata(hObject,handles)
```

```
% --- Executes on slider movement.
function SL_ColorB_Callback(hObject, eventdata, handles)

    col = handles.PM_Color.BackgroundColor;
    col(3) = round(hObject.Value)/255;
    handles.Info_Trazos{handles.Trazo_Actual}.Color = col;
    handles.PM_Color.BackgroundColor = col;
    actualizar_panel_color(col,handles,1);
    handles = guidata(hObject);
guidata(hObject,handles)

function actualizar_panel_color(col,handles,actualizar_plot)

    handles.TE_ColorR.String = round(col(1) * 255) ;
    handles.TE_ColorG.String = round(col(2) * 255) ;
    handles.TE_ColorB.String = round(col(3) * 255) ;

    handles.SL_ColorR.Value = round(col(1) * 255) ;
    handles.SL_ColorG.Value = round(col(2) * 255) ;
    handles.SL_ColorB.Value = round(col(3) * 255) ;

    if (actualizar_plot)
        handles.Info_Trazos{handles.Trazo_Actual}.Color = col;
        if ~isempty(handles.Info_Trazos{handles.Trazo_Actual}.Plot)
            handles.Info_Trazos{handles.Trazo_Actual}.Plot.Color = col;
        end
    end

guidata(handles.Tracing,handles);

% --- Executes on button press in PM_Sexo.
function PM_Sexo_Callback(hObject, eventdata, handles)

    handles.Info_Trazos{handles.Trazo_Actual}.Sexo = hObject.Value;

guidata(hObject,handles);


% --- Executes on button press in PM_Edad.
function PM_Edad_Callback(hObject, eventdata, handles)

    handles.Info_Trazos{handles.Trazo_Actual}.Edad = hObject.Value;

guidata(hObject,handles);

% --- Executes on button press in PM_Discapacidad.
function PM_Discapacidad_Callback(hObject, eventdata, handles)

    handles.Info_Trazos{handles.Trazo_Actual}.Discapacidad = hObject.Value;

guidata(hObject,handles);

function Actualizar_visualizacion(hObject, h_axes)

    axes(h_axes);
    handles = guidata(hObject);
    S = handles.Visu.Sexo;
    E = handles.Visu.Edad;
    D = handles.Visu.Disc;
    for i = 2:length(handles.Info_Trazos)
        T = handles.Info_Trazos{i};

        if (( ~S || S==6   ||  bitand(S,bitsll(1,T.Sexo-1)) ) && ...
            ( ~E || E==126 ||  bitand(E,bitsll(1,T.Edad-1)) ) && ...
            ( ~D || D==14  ||  bitand(D,bitsll(1,T.Disc-1)) ))
            handles.Info_Trazos{i}.Plot.Visible = 'On';
        else
            handles.Info_Trazos{i}.Plot.Visible = 'Off';
        end
    end
```

```
guidata (hObject,handles);

%Panel Menu
% --- Executes on button press in PB_Guardar.
function PB_Guardar_Callback(hObject, eventdata, handles)
% hObject    handle to PB_Guardar (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

    savefig(handles.output,[handles.path_proyecto '\Tracing\' ...
            handles.nombre_archivo '.fig']);

guidata(hObject,handles);

% --- Executes on button press in RB_Sexo1.
function RB_Sexo1_Callback(hObject, eventdata, handles)
    handles.Visu.Sexo = bitset(handles.Visu.Sexo,2,hObject.Value);
    guidata(hObject,handles);
    Actualizar_visualizacion(hObject,handles.Axes);
guidata(hObject,handles);

% --- Executes on button press in RB_Sexo2.
function RB_Sexo2_Callback(hObject, eventdata, handles)
    handles.Visu.Sexo = bitset(handles.Visu.Sexo,3,hObject.Value);
    guidata(hObject,handles);
    Actualizar_visualizacion(hObject,handles.Axes);
guidata(hObject,handles);

% --- Executes on button press in RB_Disc1.
function RB_Disc1_Callback(hObject, eventdata, handles)
    handles.Visu.Disc = bitset(handles.Visu.Disc,2,hObject.Value);
    guidata(hObject,handles);
    Actualizar_visualizacion(hObject,handles.Axes);
guidata(hObject,handles);

% --- Executes on button press in RB_Disc2.
function RB_Disc2_Callback(hObject, eventdata, handles)
    handles.Visu.Disc = bitset(handles.Visu.Disc,3,hObject.Value);
    guidata(hObject,handles);
    Actualizar_visualizacion(hObject,handles.Axes);
guidata(hObject,handles);

% --- Executes on button press in RB_Disc3.
function RB_Disc3_Callback(hObject, eventdata, handles)
    handles.Visu.Disc = bitset(handles.Visu.Disc,4,hObject.Value);
    guidata(hObject,handles);
    Actualizar_visualizacion(hObject,handles.Axes);
guidata(hObject,handles);

% --- Executes on button press in RB_Edad1.
function RB_Edad1_Callback(hObject, eventdata, handles)
    handles.Visu.Edad = bitset(handles.Visu.Edad,2,hObject.Value);
    guidata(hObject,handles);
    Actualizar_visualizacion(hObject,handles.Axes);
guidata(hObject,handles);

% --- Executes on button press in RB_Edad2.
function RB_Edad2_Callback(hObject, eventdata, handles)
    handles.Visu.Edad = bitset(handles.Visu.Edad,3,hObject.Value);
    guidata(hObject,handles);
    Actualizar_visualizacion(hObject,handles.Axes);
guidata(hObject,handles);

% --- Executes on button press in RB_Edad3.
function RB_Edad3_Callback(hObject, eventdata, handles)
    handles.Visu.Edad = bitset(handles.Visu.Edad,4,hObject.Value);
    guidata(hObject,handles);
    Actualizar_visualizacion(hObject,handles.Axes);
guidata(hObject,handles);

% --- Executes on button press in RB_Edad4.
function RB_Edad4_Callback(hObject, eventdata, handles)
    handles.Visu.Edad = bitset(handles.Visu.Edad,5,hObject.Value);
    guidata(hObject,handles);
```

```
    Actualizar_visualizacion(hObject,handles.Axes);
guidata(hObject,handles);

% --- Executes on button press in RB_Edad5.
function RB_Edad5_Callback(hObject, eventdata, handles)
    handles.Visu.Edad = bitset(handles.Visu.Edad,6,hObject.Value);
    guidata(hObject,handles);
    Actualizar_visualizacion(hObject,handles.Axes);
guidata(hObject,handles);

% --- Executes on button press in RB_Edad6.
function RB_Edad6_Callback(hObject, eventdata, handles)
    handles.Visu.Edad = bitset(handles.Visu.Edad,7,hObject.Value);
    guidata(hObject,handles);
    Actualizar_visualizacion(hObject,handles.Axes);
guidata(hObject,handles);

    % --- Executes on button press in PB_PDF.
function PB_PDF_Callback(hObject, eventdata, handles)

    TextDatos = [];

    TextDatos = [TextDatos 'Nombre del Proyecto: ' handles.nombre_proyecto '\n'];
    TextDatos = [TextDatos 'Nombre del Archivo: ' handles.nombre_archivo '\n\n'];
    TextDatos = [TextDatos 'Lugar de estudio: ' handles.lugar_estudio '\n'];
    TextDatos = [TextDatos 'Nombre del Usuario: ' handles.nombre_usuario '\n\n'];
    TextDatos = [TextDatos 'Condiciones climáticas: ' handles.Clima '\n'];
    TextDatos = [TextDatos 'Fecha: ' handles.TE_Dia.String '/' handles.TE_Mes.String
'/' handles.TE_Anio.String '\n'];
    TextDatos = [TextDatos 'Hora: ' handles.TE_Hora.String ':'
handles.TE_Minuto.String '\n\n'];

    TextDatos = [TextDatos 'Filtros aplicados:'];
    TextDatos = [TextDatos '\nEdad:'];
    if (handles.RB_Edad1.Value) TextDatos = [TextDatos ' ' handles.RB_Edad1.String];
end
    if (handles.RB_Edad2.Value) TextDatos = [TextDatos ' ' handles.RB_Edad2.String];
end
    if (handles.RB_Edad3.Value) TextDatos = [TextDatos ' ' handles.RB_Edad3.String];
end
    if (handles.RB_Edad4.Value) TextDatos = [TextDatos ' ' handles.RB_Edad4.String];
end
    if (handles.RB_Edad5.Value) TextDatos = [TextDatos ' ' handles.RB_Edad5.String];
end
    if (handles.RB_Edad6.Value) TextDatos = [TextDatos ' ' handles.RB_Edad6.String];
end
    if (TextDatos(end) == ':')  TextDatos = [TextDatos ' Ninguno']; end
    TextDatos = [TextDatos '\nSexo:'];
    if (handles.RB_Sexo1.Value) TextDatos = [TextDatos ' ' handles.RB_Sexo1.String];
end
    if (handles.RB_Sexo2.Value) TextDatos = [TextDatos ' ' handles.RB_Sexo2.String];
end
    if (TextDatos(end) == ':')  TextDatos = [TextDatos ' Ninguno']; end
    TextDatos = [TextDatos '\nDiscapacidad:'];
    if (handles.RB_Disc1.Value) TextDatos = [TextDatos ' ' handles.RB_Disc1.String];
end
    if (handles.RB_Disc2.Value) TextDatos = [TextDatos ' ' handles.RB_Disc2.String];
end
    if (handles.RB_Disc3.Value) TextDatos = [TextDatos ' ' handles.RB_Disc3.String];
end
    if (TextDatos(end) == ':')  TextDatos = [TextDatos ' Ninguno']; end

    % Crear la figura
    f = figure('Units', 'Pixels', 'Position', [200 30 210 300].*2);

    uicontrol(f,'Style','Text','Units', 'Pixels', 'Position', [10,280,190,10].*2,...
    'String', ['Tracing - ' handles.nombre_archivo], 'FontSize', 12,
'BackgroundColor','w');

    t = uicontrol(f,'Style','Text','Units', 'Pixels', 'Position',
[110,210,90,65].*2,...
    'FontSize', 6,  'HorizontalAlignment', 'Left',...
    'BackgroundColor','w');
```

```
    t.String = textwrap(t,{handles.Notas});

    t = uicontrol(f,'Style','Text','Units', 'Pixels', 'Position',
[10,210,90,65].*2,...
    'FontSize', 6,  'HorizontalAlignment', 'Left',...
    'BackgroundColor','w');

    t.String = textwrap(t,{sprintf(TextDatos)});

    h_axes = axes('Units', 'Pixels', 'Position', [10, 10, 190,190].*2);

    % Llenar axes
    imshow (handles.data_imagen); hold on;
    axis image;

    % Plot Figure
    for i = 2:length(handles.Info_Trazos)
        T = handles.Info_Trazos{i};
        if ( strcmp( T.Plot.Visible, 'on' ) )
            plot(h_axes, T.X, T.Y , 'Color',T.Color, 'LineWidth', 1);
        end
    end

    % Crear archivo
    pos = 1;
    while( exist([handles.path_proyecto '\Tracing\PDF\' handles.nombre_archivo '_'
num2str(pos) '.PDF'],'file'))
        pos = pos + 1;
    end

    figureName = [handles.path_proyecto '\Tracing\PDF\' handles.nombre_archivo '_'
num2str(pos) '.PDF'];
    set(gcf,'paperunits','centimeters');
    set(gcf,'papersize',[21 29.7]);
    set(gcf,'paperposition',[0,0,21,29.7]);
    print(gcf, '-dpdf',figureName);
    if ispc
        winopen(figureName);
    end
close(f);


% --- Executes on button press in PB_Cerrar.
function PB_Cerrar_Callback(hObject, eventdata, handles)

    choice = questdlg(sprintf(['¿Desea guardar el archivo antes de cerrarlo?']), ...
                            'Cerrar archivo', ...
                            'Guardar','Continuar','Guardar');
    if strcmp(choice,'Guardar')
            PB_Guardar_Callback(handles.PB_Guardar,eventdata,handles);
    end

    openfig([handles.path_proyecto '\Pantalla_Principal.fig']);

delete(handles.output);
```

```
function varargout = Mapping(varargin)
% MAPPING MATLAB code for Mapping.fig
%      MAPPING, by itself, creates a new MAPPING or raises the existing
%      singleton*.
%
%      H = MAPPING returns the handle to a new MAPPING or the handle to
%      the existing singleton*.
%
%      MAPPING('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in MAPPING.M with the given input arguments.
%
%      MAPPING('Property','Value',...) creates a new MAPPING or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before Mapping_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to Mapping_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help Mapping

% Last Modified by GUIDE v2.5 06-Dec-2016 03:00:36

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Mapping_OpeningFcn, ...
                   'gui_OutputFcn',  @Mapping_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

% --- Executes just before Mapping is made visible.
function Mapping_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to Mapping (see VARARGIN)

warning('off','all')
set(findall(handles.Mapping,'Units','Pixels'),'Units','Normalized');

handles.AppFolder = varargin{1}{1};
handles.path_proyecto = varargin{1}{2};
handles.nombre_proyecto = varargin{1}{3};
handles.nombre_archivo = varargin{1}{4};
handles.lugar_estudio = varargin{1}{5};
handles.nombre_usuario = varargin{1}{6};
handles.Mapping.Name = handles.nombre_archivo;

handles.TE_Proyecto.String = handles.nombre_proyecto;
handles.TE_Archivo.String = handles.nombre_archivo;
handles.TE_Lugar.String = handles.lugar_estudio;
handles.Clima = '';
handles.Notas = 'Ingrese sus notas';

handles.Visu.Sexo = 0;
handles.Visu.Edad = 0;
```

```
handles.Visu.Disc = 0;
handles.Tipos_marcador = {'', 'o', '+', '*','.','x','s','d','p'};
handles.Plot_Ultimo = [];
handles.Info_Actividades = cell(1);

set(findall(handles.Panel_Actividades,'Enable','On'),'Enable','Off');
set(findall(handles.Panel_Menu,'Enable','On'),'Enable','Off');

handles.PB_Cerrar.Enable = 'On';

% Choose default command line output for Mapping
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes Mapping wait for user response (see UIRESUME)
% uiwait(handles.Mapping);

% --- Outputs from this function are returned to the command line.
function varargout = Mapping_OutputFcn(hObject, eventdata, handles)

% Get default command line output from handles structure
varargout{1} = handles.output;

%--------------------------------------------------------------------------
% Panel Informacion
% --- Executes on slider movement.
function SL_Hora_Callback(hObject, eventdata, handles)

    valor = round(hObject.Value);
    handles.TE_Hora.String = num2str(valor,'%02u');

guidata(hObject, handles);

% --- Executes on slider movement.
function SL_Minuto_Callback(hObject, eventdata, handles)

    valor = round(hObject.Value);
    handles.TE_Minuto.String = num2str(valor,'%02u');

guidata(hObject, handles);

% --- Executes on slider movement.
function SL_Dia_Callback(hObject, eventdata, handles)

    valor = round(hObject.Value);
    handles.TE_Dia.String = num2str(valor,'%02u');

guidata(hObject, handles);

% --- Executes on slider movement.
function SL_Mes_Callback(hObject, eventdata, handles)

    valor = round(hObject.Value);
    handles.TE_Mes.String = num2str(valor,'%02u');

guidata(hObject, handles);

% --- Executes on slider movement.
function SL_Anio_Callback(hObject, eventdata, handles)

    valor = round(hObject.Value);
    handles.TE_Anio.String = num2str(valor,'%04u');

guidata(hObject, handles);

function TE_Clima_Callback(hObject, eventdata, handles)

    handles.Clima = hObject.String;
```

```
guidata(hObject,handles)

% --- Executes on button press in PB_Notas.
function PB_Notas_Callback(hObject, eventdata, handles)

    answer = inputdlg({'Ingrese las notas - No exceda el espacio en la ventana'},...
                      'Notas - Mapping', [12 50], {handles.Notas});
    if (isempty(answer)) return; end
    handles.Notas = answer{:};

guidata(hObject,handles);

% --- Executes on button press in PB_Importar_Imagen.
function PB_Importar_Imagen_Callback(hObject, eventdata, handles)

    image_path = uipickfiles('FilterSpec', [handles.AppFolder...
                 '\Proyectos\' handles.nombre_proyecto '\Imagenes\*.jpg'],...
                 'Prompt', 'Seleccione un archivo JPG', 'NumFiles', 1);

    if (~iscell(image_path) && ~image_path) return; end

    cla(handles.Axes)

    if (isempty(strfind(image_path{1},[handles.AppFolder...
        '\Proyectos\' handles.nombre_proyecto '\Imagenes\'])))

        copyfile(image_path{1},[handles.AppFolder '\Proyectos\'...
                 handles.nombre_proyecto '\Imagenes\Mapping_'...
                 handles.nombre_archivo '.jpg']);
        image_path{1} = [handles.AppFolder '\Proyectos\'...
                 handles.nombre_proyecto '\Imagenes\Mapping_'...
                 handles.nombre_archivo '.jpg'];
    end
    handles.data_imagen = imread( image_path{1} );
    axes(handles.Axes);
    handles.Imagen = imshow (handles.data_imagen); hold on;
    axis image;

    set(findall(handles.Panel_Actividades,'Enable','Off'),'Enable','On');
    set(findall(handles.Panel_Menu,'Enable','Off'),'Enable','On');
    set(findall(handles.Panel_Actividad,'Enable','On'),'Enable','Off');
    hObject.Enable = 'Off';


guidata(hObject,handles);

% Panel Trazos
% --- Executes on button press in PB_Agregar.
function PB_Agregar_Callback(hObject, eventdata, handles)

    Lista_Actividades = handles.PM_Actividades.String;
    if (~iscell(Lista_Actividades)) Lista_Actividades = {'Nueva'}; end
    pos = 1;
    while(1)
        if (~any(strcmp(Lista_Actividades,['Actividad ' num2str(pos)]))) break; end
        pos = pos + 1;
    end

    strText = sprintf(['Ingrese el nombre de la actividad.\n'...
                       'Use solo letras y/o números.']);
    strTitle = 'Nombre de actividad';
    nombre = ingresar_nombre({strText}, strTitle,{['Actividad ' num2str(pos)]});
    if (isempty(nombre)) return; end

    while ( any(strcmp(Lista_Actividades,nombre) ) )
        choice = questdlg(sprintf(['Esta actividad ya existe.\n' ...
                 'Si continúa los datos serán sobrescritos.']), ...
                 'Nombre de actividad en uso', ...
                 'Continuar','Ingresar otro nombre','Continuar');
            switch choice
                case 'Continuar'
                    pos = find( strcmp(Lista_Actividades,nombre) == 1);
                    handles.PM_Actividades.String(pos) = [];
```

```
                    delete(handles.Info_Actividades{pos}.Plot);
                    handles.Info_Actividades(pos) = [];
                    Lista_Actividades = handles.PM_Actividades.String;
                case 'Ingresar otro nombre'
                    nombre = ingresar_nombre(nombre);
                    if (isempty(nombre)) return; end
            end
    end
    Lista_Actividades{end+1} = nombre;
    handles.PM_Actividades.String = Lista_Actividades;
    handles.PM_Actividades.Value = length(Lista_Actividades);
    handles.Actividad_Actual = length(Lista_Actividades);
    handles.Info_Actividades{end+1} = struct('Nombre_Actividad', nombre,...
                                'Color', [0 0 0], 'Marcador', 2,...
                                'Tamano_Marcador', 6,'Sexo', 1,...
                                'Edad', 1, 'Disc', 1,...
                                'X', [], 'Y', [], 'Plot' , []);
    guidata(hObject,handles);
    PB_Modificar_Actividad_Callback(handles.PB_Modificar_Actividad, eventdata,
handles);

guidata(hObject,handles);

% --- Executes on button press in PB_Modificar_Actividad.
function PB_Modificar_Actividad_Callback(hObject, eventdata, handles)

    handles.Actividad_Actual = handles.PM_Actividades.Value;

    if (handles.Actividad_Actual == 1)
        msgbox('Seleccione una Actividad válida para editar', 'Error','error');
        return;
    end

    set(findall(handles.Panel_Actividades,'Enable','On'),'Enable','Off');
    set(findall(handles.Panel_Menu,'Enable','On'),'Enable','Off');
    set(findall(handles.Panel_Actividad,'Enable','Off'),'Enable','On');

    Actividad = handles.Info_Actividades{handles.Actividad_Actual};

    handles.PM_Color.BackgroundColor = Actividad.Color;
    guidata(hObject,handles);
    actualizar_panel_color(Actividad.Color,handles,0);
    handles.PM_Sexo.Value = Actividad.Sexo;
    handles.PM_Edad.Value = Actividad.Edad;
    handles.PM_Discapacidad.Value = Actividad.Disc;

    handles.PM_Marcador.Value = Actividad.Marcador;
    handles.TE_Tamano.String = Actividad.Tamano_Marcador;
    handles.SL_Tamano.Value = Actividad.Tamano_Marcador;

    for i = 2:length(handles.Info_Actividades)
        if ~isempty(handles.Info_Actividades{i}.Plot)
            handles.Info_Actividades{i}.Plot.Visible = 'Off';
        end
    end

    Actividad.Plot.Visible = 'On';

guidata(hObject,handles);

function PB_Eliminar_Actividad_Callback(hObject, eventdata, handles)

    handles.Actividad_Actual = handles.PM_Actividades.Value;

    if (handles.Actividad_Actual == 1)
        msgbox('Seleccione una Actividad válida para eliminar', 'Error','error');
        return;
    end

    handles.PM_Actividades.Value = 1;

    choice = questdlg(sprintf(['¿Desea eliminar la Actividad seleccionada?.\n' ...
                               'Una vez eliminada no podrá recuperarla.']), ...
```

```
                                'Eliminar Actividad', ...
                                'Eliminar','Cancelar','Cancelar');
    switch choice
        case 'Eliminar'
            handles.PM_Actividades.String(handles.Actividad_Actual) = [];
            delete(handles.Info_Actividades{handles.Actividad_Actual}.Plot);
            handles.Info_Actividades(handles.Actividad_Actual) = [];
        case 'Cancelar'
            return;
    end
    handles.Actividad_Actual = 1;
    guidata(hObject, handles);
    Actualizar_visualizacion(hObject,handles.Axes);
guidata(hObject,handles);

%Panel Menu Actividad

% --- Executes on button press in PB_Dibujar.
function PB_Dibujar_Callback(hObject, eventdata, handles)

    set(findall(handles.Panel_Actividad,'Enable','On'),'Enable','Off');
    handles.PB_Terminar.Enable = 'On';

    set(handles.Axes,'ButtonDownFcn',{@drawCurrPoint,handles.Mapping});
    set(handles.Imagen,'ButtonDownFcn',{@drawCurrPoint,handles.Mapping});

guidata(hObject, handles);

% --- Executes on button press in PB_Terminar.
function PB_Terminar_Callback(hObject, eventdata, handles)
    set(handles.Axes,'ButtonDownFcn','');
    set(handles.Imagen,'ButtonDownFcn','');
    set(findall(handles.Panel_Actividades,'Enable','Off'),'Enable','On');

guidata(hObject, handles);

function drawCurrPoint(varargin)
    handles = guidata(varargin{3});
    pt = get(handles.Axes, 'CurrentPoint');
    x = pt(1,1);
    y = pt(1,2);
    axes(handles.Axes);
    Act = handles.Info_Actividades{handles.Actividad_Actual};

    Act.X = [Act.X x];
    Act.Y = [Act.Y y];
    if isempty(Act.Plot)
        Act.Plot = plot(Act.X , Act.Y , 'LineStyle', 'none',...
            'Marker', handles.Tipos_marcador{Act.Marcador}, 'Color',Act.Color,...
            'MarkerFaceColor', Act.Color, 'MarkerSize', Act.Tamano_Marcador);
    else
        Act.Plot.XData = Act.X;
        Act.Plot.YData = Act.Y;
    end
    handles.Info_Actividades{handles.Actividad_Actual} = Act;
guidata(varargin{3}, handles);

function stopDrawPoint(varargin)
    handles = guidata(varargin{3});
    set(handles.Axes,'ButtonDownFcn','');
    set(handles.Imagen,'ButtonDownFcn','');

    set(findall(handles.Panel_Menu_Trazos,'Enable','Off'),'Enable','On');

guidata(varargin{3}, handles);

% --- Executes on button press in PB_Guardar_Actividad.
function PB_Guardar_Actividad_Callback(hObject, eventdata, handles)

    if (isempty(handles.Info_Actividades{handles.Actividad_Actual}.Plot))
        choice = questdlg(sprintf(['No se ha dibujado ningún punto.\n' ...
                                'Si continúa la actividad no será guardado.']), ...
                                'Actividad sin dibujar', ...
```

```
                                'Continuar','Cancelar','Cancelar');
        if (strcmp(choice,'Continuar'))
            handles.PM_Actividades.String(handles.Actividad_Actual) = [];
            handles.Info_Actividades(handles.Actividad_Actual) = [];
            handles.PM_Actividades.Value = 1;
        end
    else
        handles.Info_Actividades{handles.Actividad_Actual}.Color =
handles.PM_Color.BackgroundColor;
        handles.Info_Actividades{handles.Actividad_Actual}.Sexo =
handles.PM_Sexo.Value;
        handles.Info_Actividades{handles.Actividad_Actual}.Edad =
handles.PM_Edad.Value;
        handles.Info_Actividades{handles.Actividad_Actual}.Disc =
handles.PM_Discapacidad.Value;
    end
    handles.TE_Nombre_Trazo.String = '';
    handles.PM_Color.BackgroundColor = [1 1 1];

    handles.PM_Actividades.Value = 1;
    handles.Actividad_Actual = 1;

    actualizar_panel_color([1 1 1],handles,0);
    handles = guidata(hObject);
    handles.PM_Sexo.Value = 1;
    handles.PM_Edad.Value = 1;
    handles.PM_Discapacidad.Value = 1;

    set(findall(handles.Panel_Actividades,'Enable','Off'),'Enable','On');
    set(findall(handles.Panel_Actividad,'Enable','On'),'Enable','Off');
    set(findall(handles.Panel_Menu,'Enable','Off'),'Enable','On');
    guidata(hObject, handles);
    Actualizar_visualizacion(hObject,handles.Axes);

guidata(hObject, handles);

function PM_Marcador_Callback(hObject, eventdata, handles)
    if hObject.Value ~= 1
        handles.Info_Actividades{handles.Actividad_Actual}.Marcador = hObject.Value;
    end
    if ~isempty(handles.Info_Actividades{handles.Actividad_Actual}.Plot)
        handles.Info_Actividades{handles.Actividad_Actual}.Plot.Marker = ...
                handles.Tipos_marcador{hObject.Value};
    end
guidata(hObject,handles);

% --- Executes on slider movement.
function SL_Tamano_Callback(hObject, eventdata, handles)
    handles.TE_Tamano.String = round(hObject.Value,2);
    handles.Info_Actividades{handles.Actividad_Actual}.Tamano_Marcador =...
            round(hObject.Value,2);
    if ~isempty(handles.Info_Actividades{handles.Actividad_Actual}.Plot)
        handles.Info_Actividades{handles.Actividad_Actual}.Plot.MarkerSize =
hObject.Value;
    end
guidata(hObject, handles);

% --- Executes on selection change in PM_Color.
function PM_Color_Callback(hObject, eventdata, handles)
% hObject    handle to PM_Color (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

    colores = {[], [1 1 0],[1 0 1],[0 1 1],[1 0 0],[0 1 0],[0 0 1],[1 1 1],[0 0 0]};

    if (hObject.Value ~= 1)
        hObject.BackgroundColor = colores{hObject.Value};
        hObject.Value = 1;
        col = hObject.BackgroundColor;
        actualizar_panel_color(col,handles,1);
        handles = guidata(hObject);
    end
guidata(hObject,handles);
```

```
% --- Executes on slider movement.
function SL_ColorR_Callback(hObject, eventdata, handles)
    col = handles.PM_Color.BackgroundColor;
    col(1) = round(hObject.Value)/255;
    handles.Info_Actividades{handles.Actividad_Actual}.Color = col;
    handles.PM_Color.BackgroundColor = col;
    actualizar_panel_color(col,handles,1);
guidata(hObject,handles)

% --- Executes on slider movement.
function SL_ColorG_Callback(hObject, eventdata, handles)

    col = handles.PM_Color.BackgroundColor;
    col(2) = round(hObject.Value)/255;
    handles.Info_Actividades{handles.Actividad_Actual}.Color = col;
    handles.PM_Color.BackgroundColor = col;
    actualizar_panel_color(col,handles,1);

guidata(hObject,handles)

% --- Executes on slider movement.
function SL_ColorB_Callback(hObject, eventdata, handles)

    col = handles.PM_Color.BackgroundColor;
    col(3) = round(hObject.Value)/255;
    handles.Info_Actividades{handles.Actividad_Actual}.Color = col;
    handles.PM_Color.BackgroundColor = col;
    actualizar_panel_color(col,handles,1);

guidata(hObject,handles)

function actualizar_panel_color(col,handles,actualizar_plot)

    handles.TE_ColorR.String = round(col(1) * 255) ;
    handles.TE_ColorG.String = round(col(2) * 255) ;
    handles.TE_ColorB.String = round(col(3) * 255) ;

    handles.SL_ColorR.Value = round(col(1) * 255) ;
    handles.SL_ColorG.Value = round(col(2) * 255) ;
    handles.SL_ColorB.Value = round(col(3) * 255) ;

    if (actualizar_plot)
        handles.Info_Actividades{handles.Actividad_Actual}.Color = col;
        if ~isempty(handles.Info_Actividades{handles.Actividad_Actual}.Plot)
            handles.Info_Actividades{handles.Actividad_Actual}.Plot.MarkerFaceColor =
col;
            handles.Info_Actividades{handles.Actividad_Actual}.Plot.Color = col;
        end
    end
guidata(handles.Mapping,handles)

% --- Executes on button press in PM_Sexo.
function PM_Sexo_Callback(hObject, eventdata, handles)

    handles.Info_Actividades{handles.Actividad_Actual}.Sexo = hObject.Value;

guidata(hObject,handles);


% --- Executes on button press in PM_Edad.
function PM_Edad_Callback(hObject, eventdata, handles)

    handles.Info_Actividades{handles.Actividad_Actual}.Edad = hObject.Value;

guidata(hObject,handles);

% --- Executes on button press in PM_Discapacidad.
function PM_Discapacidad_Callback(hObject, eventdata, handles)

    handles.Info_Actividades{handles.Actividad_Actual}.Discapacidad = hObject.Value;

guidata(hObject,handles);
```

```
% --- Executes on button press in PB_Iniciar_Eliminar_Puntos.
function PB_Iniciar_Eliminar_Puntos_Callback(hObject, eventdata, handles)

    Act = handles.Info_Actividades{handles.Actividad_Actual};


    if strcmp(hObject.String,'Iniciar')
        if (isempty(Act.Plot)||isempty(Act.Plot.XData)) return; end
        handles.SL_Ultimo_Punto.Max    = length(Act.Plot.XData);
        handles.SL_Ultimo_Punto.SliderStep = [1 1].*(1/(length(Act.Plot.XData)-1));
        handles.SL_Ultimo_Punto.Value = length(Act.Plot.XData);
        handles.TE_Ultimo_Punto.String = round(length(Act.Plot.XData),0);
        handles.Plot_Ultimo = [];
        guidata(hObject,handles);
        Actualizar_Ultimo_Punto(handles);
        handles = guidata(hObject);
        hObject.String = 'Terminar';
        set(findall(handles.Panel_Actividad,'Enable','On'),'Enable','Off');
        set(findall(handles.Panel_Puntos,'Enable','Off'),'Enable','On');
    else

        delete(handles.Plot_Ultimo);
        hObject.String = 'Iniciar';
        set(findall(handles.Panel_Actividad,'Enable','Off'),'Enable','On');
        set(findall(handles.Panel_Puntos,'Enable','On'),'Enable','Off');
        hObject.Enable = 'On';
    end
guidata(hObject,handles);


% --- Executes on slider movement.
function SL_Ultimo_Punto_Callback(hObject, eventdata, handles)
    handles.TE_Ultimo_Punto.String = round(hObject.Value,0);
    guidata(hObject,handles);
    Actualizar_Ultimo_Punto(handles);
    handles = guidata(hObject);
guidata(hObject,handles);

function Actualizar_Ultimo_Punto(handles)
    Act = handles.Info_Actividades{handles.Actividad_Actual};
    pos = str2double(handles.TE_Ultimo_Punto.String);

    if isempty(handles.Plot_Ultimo)
        handles.Plot_Ultimo = plot(Act.Plot.XData(pos), Act.Plot.YData(pos),
'LineStyle', 'none',...
            'Marker', handles.Tipos_marcador{Act.Marcador}, 'Color',Act.Color,...
            'MarkerFaceColor', 1-Act.Color, 'MarkerSize', 2*Act.Tamano_Marcador);
    else
        handles.Plot_Ultimo.XData = Act.Plot.XData(pos);
        handles.Plot_Ultimo.YData = Act.Plot.YData(pos);
    end
guidata(handles.Mapping,handles);

function PB_Eliminar_Punto_Callback(hObject, eventdata, handles)

    Act = handles.Info_Actividades{handles.Actividad_Actual};
    pos = str2double(handles.TE_Ultimo_Punto.String);
    Act.Plot.XData(pos) = [];
    Act.Plot.YData(pos) = [];
    Act.X(pos) = [];
    Act.Y(pos) = [];
    if isempty(Act.Plot.XData)
        handles.Info_Actividades{handles.Actividad_Actual} = Act;

PB_Iniciar_Eliminar_Puntos_Callback(handles.PB_Iniciar_Eliminar_Puntos,eventdata,hand
les);
        guidata(hObject,handles);
        return;
    end

    handles.SL_Ultimo_Punto.Value = min(handles.SL_Ultimo_Punto.Value,...
                                        length(Act.Plot.XData));
    handles.TE_Ultimo_Punto.String = round(handles.SL_Ultimo_Punto.Value,0);
```

```
    handles.SL_Ultimo_Punto.Max    = length(Act.Plot.XData);
    if handles.SL_Ultimo_Punto.Max == 1
        aux = 0;
    else
        aux = 1/(length(Act.Plot.XData)-1);
    end
    handles.SL_Ultimo_Punto.SliderStep = [1 1].*aux;

    handles.Info_Actividades{handles.Actividad_Actual} = Act;

    guidata(hObject,handles);
    Actualizar_Ultimo_Punto(handles);
    handles = guidata(hObject);
guidata(hObject,handles);

function Actualizar_visualizacion(hObject, h_axes)

    axes(h_axes);
    handles = guidata(hObject);
    S = handles.Visu.Sexo;
    E = handles.Visu.Edad;
    D = handles.Visu.Disc;
    hand_leg = [];
    name_leg = {};
    for i = 2:length(handles.Info_Actividades)
        T = handles.Info_Actividades{i};

        if (( ~S || S==6   ||  bitand(S,bitsll(1,T.Sexo-1)) ) && ...
            ( ~E || E==126 ||  bitand(E,bitsll(1,T.Edad-1)) ) && ...
            ( ~D || D==14  ||  bitand(D,bitsll(1,T.Disc-1)) ))
            handles.Info_Actividades{i}.Plot.Visible = 'On';
            hand_leg = [hand_leg handles.Info_Actividades{i}.Plot];
            name_leg = [name_leg handles.Info_Actividades{i}.Nombre_Actividad];
        else
            handles.Info_Actividades{i}.Plot.Visible = 'Off';
        end
    end
    if isempty(name_leg)
        legend('hide');
    else
        legend(hand_leg,'String', name_leg, 'Location', 'eastoutside');
    end
guidata (hObject,handles);

%Panel Menu
% --- Executes on button press in PB_Guardar_Actividad.
function PB_Guardar_Callback(hObject, eventdata, handles)
% hObject    handle to PB_Guardar_Actividad (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

    savefig(handles.output,[handles.path_proyecto '\Mapping\' ...
            handles.nombre_archivo '.fig']);

guidata(hObject,handles);

% --- Executes on button press in RB_Sexo1.
function RB_Sexo1_Callback(hObject, eventdata, handles)
    handles.Visu.Sexo = bitset(handles.Visu.Sexo,2,hObject.Value);
    guidata(hObject,handles);
    Actualizar_visualizacion(hObject,handles.Axes);
guidata(hObject,handles);

% --- Executes on button press in RB_Sexo2.
function RB_Sexo2_Callback(hObject, eventdata, handles)
    handles.Visu.Sexo = bitset(handles.Visu.Sexo,3,hObject.Value);
    guidata(hObject,handles);
    Actualizar_visualizacion(hObject,handles.Axes);
guidata(hObject,handles);

% --- Executes on button press in RB_Disc1.
function RB_Disc1_Callback(hObject, eventdata, handles)
    handles.Visu.Disc = bitset(handles.Visu.Disc,2,hObject.Value);
    guidata(hObject,handles);
```

```
    Actualizar_visualizacion(hObject,handles.Axes);
guidata(hObject,handles);

% --- Executes on button press in RB_Disc2.
function RB_Disc2_Callback(hObject, eventdata, handles)
    handles.Visu.Disc = bitset(handles.Visu.Disc,3,hObject.Value);
    guidata(hObject,handles);
    Actualizar_visualizacion(hObject,handles.Axes);
guidata(hObject,handles);

% --- Executes on button press in RB_Disc3.
function RB_Disc3_Callback(hObject, eventdata, handles)
    handles.Visu.Disc = bitset(handles.Visu.Disc,4,hObject.Value);
    guidata(hObject,handles);
    Actualizar_visualizacion(hObject,handles.Axes);
guidata(hObject,handles);

% --- Executes on button press in RB_Edad1.
function RB_Edad1_Callback(hObject, eventdata, handles)
    handles.Visu.Edad = bitset(handles.Visu.Edad,2,hObject.Value);
    guidata(hObject,handles);
    Actualizar_visualizacion(hObject,handles.Axes);
guidata(hObject,handles);

% --- Executes on button press in RB_Edad2.
function RB_Edad2_Callback(hObject, eventdata, handles)
    handles.Visu.Edad = bitset(handles.Visu.Edad,3,hObject.Value);
    guidata(hObject,handles);
    Actualizar_visualizacion(hObject,handles.Axes);
guidata(hObject,handles);

% --- Executes on button press in RB_Edad3.
function RB_Edad3_Callback(hObject, eventdata, handles)
    handles.Visu.Edad = bitset(handles.Visu.Edad,4,hObject.Value);
    guidata(hObject,handles);
    Actualizar_visualizacion(hObject,handles.Axes);
guidata(hObject,handles);

% --- Executes on button press in RB_Edad4.
function RB_Edad4_Callback(hObject, eventdata, handles)
    handles.Visu.Edad = bitset(handles.Visu.Edad,5,hObject.Value);
    guidata(hObject,handles);
    Actualizar_visualizacion(hObject,handles.Axes);
guidata(hObject,handles);

% --- Executes on button press in RB_Edad5.
function RB_Edad5_Callback(hObject, eventdata, handles)
    handles.Visu.Edad = bitset(handles.Visu.Edad,6,hObject.Value);
    guidata(hObject,handles);
    Actualizar_visualizacion(hObject,handles.Axes);
guidata(hObject,handles);

% --- Executes on button press in RB_Edad6.
function RB_Edad6_Callback(hObject, eventdata, handles)
    handles.Visu.Edad = bitset(handles.Visu.Edad,7,hObject.Value);
    guidata(hObject,handles);
    Actualizar_visualizacion(hObject,handles.Axes);
guidata(hObject,handles);

    % --- Executes on button press in PB_PDF.
function PB_PDF_Callback(hObject, eventdata, handles)

    TextDatos = [];

    TextDatos = [TextDatos 'Nombre del Proyecto: ' handles.nombre_proyecto '\n'];
    TextDatos = [TextDatos 'Nombre del Archivo: ' handles.nombre_archivo '\n\n'];
    TextDatos = [TextDatos 'Lugar de estudio: ' handles.lugar_estudio '\n'];
    TextDatos = [TextDatos 'Nombre del Usuario: ' handles.nombre_usuario '\n\n'];
    TextDatos = [TextDatos 'Condiciones climáticas: ' handles.Clima '\n'];
    TextDatos = [TextDatos 'Fecha: ' handles.TE_Dia.String '/' handles.TE_Mes.String
'/' handles.TE_Anio.String '\n'];
    TextDatos = [TextDatos 'Hora: ' handles.TE_Hora.String ':'
handles.TE_Minuto.String '\n\n'];
```

```
    TextDatos = [TextDatos 'Filtros aplicados:'];
    TextDatos = [TextDatos '\nEdad:'];
    if (handles.RB_Edad1.Value) TextDatos = [TextDatos ' ' handles.RB_Edad1.String];
end
    if (handles.RB_Edad2.Value) TextDatos = [TextDatos ' ' handles.RB_Edad2.String];
end
    if (handles.RB_Edad3.Value) TextDatos = [TextDatos ' ' handles.RB_Edad3.String];
end
    if (handles.RB_Edad4.Value) TextDatos = [TextDatos ' ' handles.RB_Edad4.String];
end
    if (handles.RB_Edad5.Value) TextDatos = [TextDatos ' ' handles.RB_Edad5.String];
end
    if (handles.RB_Edad6.Value) TextDatos = [TextDatos ' ' handles.RB_Edad6.String];
end
    if (TextDatos(end) == ':')  TextDatos = [TextDatos ' Ninguno']; end
    TextDatos = [TextDatos '\nSexo:'];
    if (handles.RB_Sexo1.Value) TextDatos = [TextDatos ' ' handles.RB_Sexo1.String];
end
    if (handles.RB_Sexo2.Value) TextDatos = [TextDatos ' ' handles.RB_Sexo2.String];
end
    if (TextDatos(end) == ':')  TextDatos = [TextDatos ' Ninguno']; end
    TextDatos = [TextDatos '\nDiscapacidad:'];
    if (handles.RB_Disc1.Value) TextDatos = [TextDatos ' ' handles.RB_Disc1.String];
end
    if (handles.RB_Disc2.Value) TextDatos = [TextDatos ' ' handles.RB_Disc2.String];
end
    if (handles.RB_Disc3.Value) TextDatos = [TextDatos ' ' handles.RB_Disc3.String];
end
    if (TextDatos(end) == ':')  TextDatos = [TextDatos ' Ninguno']; end
   % Crear la figura
    f = figure('Units', 'Pixels', 'Position', [200 30 210 300].*2);

    uicontrol(f,'Style','Text','Units', 'Pixels', 'Position', [10,280,190,10].*2,...
    'String', ['Mapping - ' handles.nombre_archivo], 'FontSize', 12,
'BackgroundColor','w');

    t = uicontrol(f,'Style','Text','Units', 'Pixels', 'Position',
[110,210,90,65].*2,...
    'FontSize', 6,  'HorizontalAlignment', 'Left',...
    'BackgroundColor','w');

    t.String = textwrap(t,{handles.Notas});

    t = uicontrol(f,'Style','Text','Units', 'Pixels', 'Position',
[10,210,90,65].*2,...
    'FontSize', 6,  'HorizontalAlignment', 'Left',...
    'BackgroundColor','w');

    t.String = textwrap(t,{sprintf(TextDatos)});

    h_axes = axes('Units', 'Pixels', 'Position', [10, 10, 190,190].*2);

    % Llenar axes
    imshow (handles.data_imagen); hold on;
    axis image;
    hand_leg = [];
    name_leg = {};

    % Plot Figure
    for i = 2:length(handles.Info_Actividades)
        Act = handles.Info_Actividades{i};
        if ( strcmp( Act.Plot.Visible, 'on' ) )
            plot(Act.X , Act.Y , 'LineStyle', 'none',...
            'Marker', handles.Tipos_marcador{Act.Marcador}, 'Color',Act.Color,...
            'MarkerFaceColor', Act.Color, 'MarkerSize', Act.Tamano_Marcador);
            hand_leg = [hand_leg handles.Info_Actividades{i}.Plot];
            name_leg = [name_leg handles.Info_Actividades{i}.Nombre_Actividad];
        end
    end

    legend(h_axes, hand_leg,'String', name_leg, 'Location', 'eastoutside');
    % Crear archivo
    pos = 1;
    while( exist([handles.path_proyecto '\Mapping\PDF\' handles.nombre_archivo '_'
num2str(pos) '.PDF'],'file'))
```

```
        pos = pos + 1;
    end

    figureName = [handles.path_proyecto '\Mapping\PDF\' handles.nombre_archivo '_'
num2str(pos) '.PDF'];
    set(gcf,'paperunits','centimeters');
    set(gcf,'papersize',[21 29.7]);
    set(gcf,'paperposition',[0,0,21,29.7]);
    print(gcf, '-dpdf',figureName);
    if ispc
        winopen(figureName);
    end
close(f);


% --- Executes on button press in PB_Cerrar.
function PB_Cerrar_Callback(hObject, eventdata, handles)

    choice = questdlg(sprintf(['¿Desea guardar el archivo antes de cerrarlo?']), ...
                              'Cerrar archivo', ...
                              'Guardar','Continuar','Guardar');
    if strcmp(choice,'Guardar')
            PB_Guardar_Callback(handles.PB_Guardar_Actividad,eventdata,handles);
    end

    openfig([handles.path_proyecto '\Pantalla_Principal.fig']);

delete(handles.output);
```

```
function varargout = Tracking(varargin)
% TRACKING MATLAB code for Tracking.fig
%      TRACKING, by itself, creates a new TRACKING or raises the existing
%      singleton*.
%
%      H = TRACKING returns the handle to a new TRACKING or the handle to
%      the existing singleton*.
%
%      TRACKING('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in TRACKING.M with the given input arguments.
%
%      TRACKING('Property','Value',...) creates a new TRACKING or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before Tracking_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to Tracking_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help Tracking

% Last Modified by GUIDE v2.5 19-May-2016 09:19:36

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Tracking_OpeningFcn, ...
                   'gui_OutputFcn',  @Tracking_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

% --- Executes just before Tracking is made visible.
function Tracking_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to Tracking (see VARARGIN)

warning('off','all')
set(findall(handles.Tracking,'Units','Pixels'),'Units','Normalized');

handles.AppFolder = varargin{1}{1};
handles.path_proyecto = varargin{1}{2};
handles.nombre_proyecto = varargin{1}{3};
handles.nombre_archivo = varargin{1}{4};
handles.lugar_estudio = varargin{1}{5};
handles.nombre_usuario = varargin{1}{6};
handles.Tracking.Name = handles.nombre_archivo;

handles.TE_Proyecto.String = handles.nombre_proyecto;
handles.TE_Archivo.String = handles.nombre_archivo;
handles.TE_Lugar.String = handles.lugar_estudio;
handles.Clima = '';
handles.Notas = 'Ingrese sus notas';

handles.Visu.Sexo = 0;
handles.Visu.Edad = 0;
```

```
handles.Visu.Disc = 0;

handles.Ruta = struct('Color', [0 0 0], 'Sexo', 1,...
                      'Edad', 1, 'Disc', 1,...
                      'X', [], 'Y', [], 'Plot' , []);

handles.Registros = cell(0);

set(findall(handles.Panel_Registros,'Enable','On'),'Enable','Off');
set(findall(handles.Panel_Menu_Trazo,'Enable','On'),'Enable','Off');
set(findall(handles.Panel_Menu,'Enable','On'),'Enable','Off');

handles.PB_Cerrar.Enable = 'On';

% Choose default command line output for Tracking
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes Tracking wait for user response (see UIRESUME)
% uiwait(handles.Tracking);

% --- Outputs from this function are returned to the command line.
function varargout = Tracking_OutputFcn(hObject, eventdata, handles)

% Get default command line output from handles structure
varargout{1} = handles.output;

%--------------------------------------------------------------------------
% Panel Informacion
% --- Executes on slider movement.
function SL_Hora_Callback(hObject, eventdata, handles)

    valor = round(hObject.Value);
    handles.TE_Hora.String = num2str(valor,'%02u');

guidata(hObject, handles);

% --- Executes on slider movement.
function SL_Minuto_Callback(hObject, eventdata, handles)

    valor = round(hObject.Value);
    handles.TE_Minuto.String = num2str(valor,'%02u');

guidata(hObject, handles);

% --- Executes on slider movement.
function SL_Dia_Callback(hObject, eventdata, handles)

    valor = round(hObject.Value);
    handles.TE_Dia.String = num2str(valor,'%02u');

guidata(hObject, handles);

% --- Executes on slider movement.
function SL_Mes_Callback(hObject, eventdata, handles)

    valor = round(hObject.Value);
    handles.TE_Mes.String = num2str(valor,'%02u');

guidata(hObject, handles);

% --- Executes on slider movement.
function SL_Anio_Callback(hObject, eventdata, handles)

    valor = round(hObject.Value);
    handles.TE_Anio.String = num2str(valor,'%04u');

guidata(hObject, handles);
```

```
function TE_Clima_Callback(hObject, eventdata, handles)

    handles.Clima = hObject.String;

guidata(hObject,handles)

% --- Executes on button press in PB_Notas.
function PB_Notas_Callback(hObject, eventdata, handles)

    answer = inputdlg({'Ingrese las notas - No exceda el espacio en la ventana'},...
                      'Notas - Tracking', [12 50], {handles.Notas});
    if (isempty(answer)) return; end
    handles.Notas = answer{:};

guidata(hObject,handles);

% --- Executes on button press in PB_Importar_Imagen.
function PB_Importar_Imagen_Callback(hObject, eventdata, handles)

    image_path = uipickfiles('FilterSpec', [handles.AppFolder...
                 '\Proyectos\' handles.nombre_proyecto '\Imagenes\*.jpg'],...
                 'Prompt', 'Seleccione un archivo JPG', 'NumFiles', 1);

    if (~iscell(image_path) && ~image_path) return; end

    cla(handles.Axes)

    if (isempty(strfind(image_path{1},[handles.AppFolder...
        '\Proyectos\' handles.nombre_proyecto '\Imagenes\'])))

        copyfile(image_path{1},[handles.AppFolder '\Proyectos\'...
                 handles.nombre_proyecto '\Imagenes\Tracing_'...
                 handles.nombre_archivo '.jpg']);
        image_path{1} = [handles.AppFolder '\Proyectos\'...
                 handles.nombre_proyecto '\Imagenes\Tracing_'...
                 handles.nombre_archivo '.jpg'];
    end
    handles.data_imagen = imread( image_path{1} );
    axes(handles.Axes);
    handles.Imagen = imshow (handles.data_imagen); hold on;
    axis image;

    set(findall(handles.Panel_Menu_Trazo,'Enable','Off'),'Enable','On');
    set(findall(handles.Panel_Menu,'Enable','Off'),'Enable','On');
    hObject.Enable = 'Off';

guidata(hObject,handles);

% Panel Trazos
% --- Executes on button press in PB_Agregar_Trazo.
function PB_Agregar_Trazo_Callback(hObject, eventdata, handles)

    Lista_Trazos = handles.PM_Trazos.String;
    if (~iscell(Lista_Trazos)) Lista_Trazos = {'Nuevo Trazo'}; end
    pos = 1;
    while(1)
        if (~any(strcmp(Lista_Trazos,['Trazo ' num2str(pos)]))) break; end
        pos = pos + 1;
    end

    strText = sprintf(['Ingrese el nombre del nuevo trazo.\n'...
                       'Use solo letras y/o números.']);
    strTitle = 'Nombre de trazo';
    nombre = ingresar_nombre({strText}, strTitle,{['Trazo ' num2str(pos)]});

    if (isempty(nombre)) return; end

    while ( any(strcmp(Lista_Trazos,nombre) ) )
        choice = questdlg(sprintf(['Este nombre de trazo ya existe.\n' ...
                 'Si continúa el trazo será sobrescrito.']), ...
                 'Nombre de trazo existente', ...
                 'Continuar','Ingresar otro nombre','Continuar');
```

```
            switch choice
                case 'Continuar'
                    pos = find( strcmp(Lista_Trazos,nombre) == 1);
                    handles.PM_Trazos.String(pos) = [];
                    delete(handles.Info_Trazos{pos}.Plot);
                    handles.Info_Trazos(pos) = [];
                    Lista_Trazos = handles.PM_Trazos.String;
                case 'Ingresar otro nombre'
                    nombre = ingresar_nombre(nombre);
                    if (isempty(nombre)) return; end
            end
    end
    Lista_Trazos{end+1} = nombre;
    handles.PM_Trazos.String = Lista_Trazos;
    handles.PM_Trazos.Value = length(Lista_Trazos);
    handles.Trazo_Actual = length(Lista_Trazos);
    handles.Info_Trazos{end+1} = struct('Nombre_Trazo', nombre,...
                                'Color', [0 0 0], 'Sexo', 1,...
                                'Edad', 1, 'Disc', 1,...
                                'X', [], 'Y', [], 'Plot' , []);
    guidata(hObject,handles);
    PB_Modificar_Trazo_Callback(handles.PB_Modificar_Trazo, eventdata, handles);
    handles = guidata(hObject);
guidata(hObject,handles);

% --- Executes on button press in PB_Duplicar_Trazo.
function PB_Duplicar_Trazo_Callback(hObject, eventdata, handles)

    pos = handles.PM_Trazos.Value;
    Lista_Trazos = handles.PM_Trazos.String;
    if (pos == 1)
        msgbox('Seleccione un trazo válido', 'Error','error');
        return;
    end

    strText = sprintf(['Ingrese el nombre del nuevo trazo.\n'...
                       'Use solo letras y/o números.']);
    strTitle = 'Nombre de trazo';
    nombre = ingresar_nombre({strText}, strTitle,{[Lista_Trazos{pos} ' 2']});

    if (isempty(nombre)) return; end

    while ( any(strcmp(Lista_Trazos,nombre) ) )
        choice = questdlg(sprintf(['El nombre ingresado ya existe.\n' ...
                 'Si continúa el trazo será sobrescrito.'...
                 'Elija un nombre distinto al trazo que desea duplicar']),...
                 'Nombre de trazo existente', ...
                 'Continuar','Ingresar otro nombre','Continuar');
            if (find( strcmp(Lista_Trazos,nombre) == 1) == pos)
                choice = 'Ingresar otro nombre';
            end
            switch choice
                case 'Continuar'

                    pos_borrar = find( strcmp(Lista_Trazos,nombre) == 1);
                    handles.PM_Trazos.String(pos_borrar) = [];
                    delete(handles.Info_Trazos{pos_borrar}.Plot);
                    handles.Info_Trazos(pos_borrar) = [];
                    Lista_Trazos = handles.PM_Trazos.String;
                case 'Ingresar otro nombre'
                    nombre = ingresar_nombre(nombre);
                    if (isempty(nombre)) return; end
            end
    end

    Lista_Trazos{end+1} = nombre;
    handles.PM_Trazos.String = Lista_Trazos;
    handles.PM_Trazos.Value = length(Lista_Trazos);

    handles.Info_Trazos{end+1} = handles.Info_Trazos{pos};
    handles.Info_Trazos{end}.Nombre_Trazo = nombre;

    handles.Trazo_Actual = length(Lista_Trazos);
```

```
    handles.Info_Trazos{handles.Trazo_Actual}.Plot = ...
        plot(handles.Info_Trazos{handles.Trazo_Actual}.X ,...
             handles.Info_Trazos{handles.Trazo_Actual}.Y ,...
             'Color',handles.PM_Color.BackgroundColor, 'LineWidth', 2);

    guidata(hObject,handles);
    PB_Modificar_Trazo_Callback(handles.PB_Modificar_Trazo, eventdata, handles);
    handles = guidata(hObject);

guidata(hObject,handles);

% --- Executes on button press in PB_Modificar_Trazo.
function PB_Modificar_Trazo_Callback(hObject, eventdata, handles)

    handles.Trazo_Actual = handles.PM_Trazos.Value;

    if (handles.Trazo_Actual == 1)
        msgbox('Seleccione un Trazo válido para modificar', 'Error','error');
        return;
    end

    set(findall(handles.Panel_Menu_Trazo,'Enable','Off'),'Enable','On');
    set(findall(handles.Panel_Trazos,'Enable','On'),'Enable','Off');
    set(findall(handles.Panel_Menu,'Enable','On'),'Enable','Off');

    Trazo = handles.Info_Trazos{handles.Trazo_Actual};

    handles.TE_Nombre_Trazo.String = Trazo.Nombre_Trazo;
    handles.PM_Color.BackgroundColor = Trazo.Color;
    guidata(hObject,handles);
    actualizar_panel_color(Trazo.Color,handles,0);
    handles = guidata(hObject);
    handles.PM_Sexo.Value = Trazo.Sexo;
    handles.PM_Edad.Value = Trazo.Edad;
    handles.PM_Discapacidad.Value = Trazo.Disc;

    for i = 2:length(handles.Info_Trazos)
        handles.Info_Trazos{i}.Plot.Visible = 'Off';
    end

    Trazo.Plot.Visible = 'On';

guidata(hObject,handles);

function PB_Eliminar_Trazo_Callback(hObject, eventdata, handles)

    handles.Trazo_Actual = handles.PM_Trazos.Value;
    handles.PM_Trazos.Value = 1;

    if (handles.Trazo_Actual == 1)
        msgbox('Seleccione un Trazo válido para eliminar', 'Error','error');
        return;
    end

    choice = questdlg(sprintf(['¿Desea eliminar el Trazo seleccionado?.\n' ...
                               'Una vez eliminado no podrá recuperarlo.']), ...
                               'Eliminar Trazo', ...
                               'Eliminar','Cancelar','Cancelar');
    switch choice
        case 'Eliminar'
            handles.PM_Trazos.String(handles.Trazo_Actual) = [];
            delete(handles.Info_Trazos{handles.Trazo_Actual}.Plot);
            handles.Info_Trazos(handles.Trazo_Actual) = [];
        case 'Cancelar'
            return;
    end
    handles.Trazo_Actual = 1;

guidata(hObject,handles);

%Panel Menu Trazo
% --- Executes on button press in PB_Dibujar_Trazo.
```

```
function PB_Dibujar_Trazo_Callback(hObject, eventdata, handles)

    if(~isempty(handles.Ruta.Plot))

        delete(handles.Ruta.Plot);

        handles.Ruta.Plot = [];
        handles.Ruta.X = [];
        handles.Ruta.Y = [];
    end

    set(findall(handles.Panel_Menu_Trazo,'Enable','On'),'Enable','Off');

    set(handles.Axes,'ButtonDownFcn',{@startDragFcn,handles.Tracking});
    set(handles.Imagen,'ButtonDownFcn',{@startDragFcn,handles.Tracking});

guidata(hObject, handles);

function startDragFcn(varargin)
    handles = guidata(varargin{3});
    set( handles.Tracking, 'WindowButtonMotionFcn', {@draggingFcn,handles.Tracking}
);
    set( handles.Tracking, 'WindowButtonUpFcn', {@stopDragFcn,handles.Tracking});
    pt = get(handles.Axes, 'CurrentPoint');
    x = pt(1,1);
    y = pt(1,2);
    axes(handles.Axes);
    handles.Ruta.X = x;
    handles.Ruta.Y = y;

    handles.Ruta.Plot = plot(handles.Ruta.X, handles.Ruta.Y ,...
            'Color',handles.PM_Color.BackgroundColor, 'LineWidth', 2);

guidata(varargin{3}, handles);

function draggingFcn(varargin)
    handles = guidata(varargin{3});
    pt = get(handles.Axes, 'CurrentPoint');

    x = pt(1,1);
    y = pt(1,2);

    handles.Ruta.X = [handles.Ruta.X x];
    handles.Ruta.Y = [handles.Ruta.Y y];
    handles.Ruta.Plot.XData = handles.Ruta.X;
    handles.Ruta.Plot.YData = handles.Ruta.Y;

guidata(varargin{3}, handles);

function stopDragFcn(varargin)
    handles = guidata(varargin{3});
    set(handles.Tracking, 'WindowButtonMotionFcn', '');
    set(handles.Tracking, 'WindowButtonUpFcn', '');
    set(handles.Axes,'ButtonDownFcn','');
    set(handles.Imagen,'ButtonDownFcn','');

    set(findall(handles.Panel_Menu_Trazo,'Enable','Off'),'Enable','On');

guidata(varargin{3}, handles);

% --- Executes on button press in PB_Editar_Registro.
function PB_Editar_Registro_Callback(hObject, eventdata, handles)

    set(findall(handles.Panel_Menu_Trazo,'Enable','On'),'Enable','Off');
    set(findall(handles.Panel_Menu,'Enable','On'),'Enable','Off');
    set(findall(handles.Panel_Registros,'Enable','Off'),'Enable','On');

    SL_Registros_Callback(handles.SL_Registros, eventdata, handles);
        handles=guidata(hObject);

guidata(hObject, handles);
```

```
% --- Executes on selection change in PM_Color.
function PM_Color_Callback(hObject, eventdata, handles)
% hObject    handle to PM_Color (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

    colores = {[], [1 1 0],[1 0 1],[0 1 1],[1 0 0],[0 1 0],[0 0 1],[1 1 1],[0 0 0]};

    if (hObject.Value ~= 1)
        hObject.BackgroundColor = colores{hObject.Value};
        hObject.Value = 1;
        col = hObject.BackgroundColor;
        actualizar_panel_color(col,handles,1);
        handles = guidata(hObject);
    end
guidata(hObject,handles);

% --- Executes on slider movement.
function SL_ColorR_Callback(hObject, eventdata, handles)
    col = handles.PM_Color.BackgroundColor;
    col(1) = round(hObject.Value)/255;
    handles.Ruta.Color = col;
    handles.PM_Color.BackgroundColor = col;
    actualizar_panel_color(col,handles,1);
    handles = guidata(hObject);
guidata(hObject,handles)

% --- Executes on slider movement.
function SL_ColorG_Callback(hObject, eventdata, handles)

    col = handles.PM_Color.BackgroundColor;
    col(2) = round(hObject.Value)/255;
    handles.Ruta.Color = col;
    handles.PM_Color.BackgroundColor = col;
    actualizar_panel_color(col,handles,1);
    handles = guidata(hObject);
guidata(hObject,handles)

% --- Executes on slider movement.
function SL_ColorB_Callback(hObject, eventdata, handles)

    col = handles.PM_Color.BackgroundColor;
    col(3) = round(hObject.Value)/255;
    handles.Ruta.Color = col;
    handles.PM_Color.BackgroundColor = col;
    actualizar_panel_color(col,handles,1);
    handles = guidata(hObject);
guidata(hObject,handles)

function actualizar_panel_color(col,handles,actualizar_plot)

    handles.TE_ColorR.String = round(col(1) * 255) ;
    handles.TE_ColorG.String = round(col(2) * 255) ;
    handles.TE_ColorB.String = round(col(3) * 255) ;

    handles.SL_ColorR.Value = round(col(1) * 255) ;
    handles.SL_ColorG.Value = round(col(2) * 255) ;
    handles.SL_ColorB.Value = round(col(3) * 255) ;

    if (actualizar_plot)
        handles.Ruta.Color = col;
        if ~isempty(handles.Ruta.Plot)
            handles.Ruta.Plot.Color = col;
        end
        for i = 1:length(handles.Registros)
            if ~isempty(handles.Registros{i}.Plot)
                handles.Registros{i}.Plot.Color = col;
                handles.Registros{i}.Plot.MarkerFaceColor = col;
            end
        end
    end

guidata(handles.Tracking,handles);
```

```
% --- Executes on button press in PM_Sexo.
function PM_Sexo_Callback(hObject, eventdata, handles)

    handles.Ruta.Sexo = hObject.Value;

guidata(hObject,handles);


% --- Executes on button press in PM_Edad.
function PM_Edad_Callback(hObject, eventdata, handles)

    handles.Ruta.Edad = hObject.Value;

guidata(hObject,handles);

% --- Executes on button press in PM_Discapacidad.
function PM_Discapacidad_Callback(hObject, eventdata, handles)

    handles.Ruta.Discapacidad = hObject.Value;

guidata(hObject,handles);

%Panel Registros
% --- Executes on button press in PB_Nuevo_Registro.
function PB_Nuevo_Registro_Callback(hObject, eventdata, handles)

    N = length(handles.Registros);

    handles.Registros{N+1} = struct( 'X', [], 'Y', [], 'Plot', [],...
            'Texto', ['Registro'], 'TextPlot', []);

    handles.SL_Registros.Max = N+1;
    handles.SL_Registros.Value = N+1;
    handles.SL_Registros.SliderStep = [1 1].*1/(N+1);

    SL_Registros_Callback(handles.SL_Registros, eventdata, handles);
    handles=guidata(hObject);
guidata(hObject,handles)


% --- Executes on slider movement.
function SL_Registros_Callback(hObject, eventdata, handles)

    pos = hObject.Value;

    handles.TE_Registros.String = round(pos,0);
    if ~hObject.Value
        handles.TE_Texto_Registro.String = '';
        return;
    end

    handles.TE_Texto_Registro.String = handles.Registros{pos}.Texto;

    for i = 1:length(handles.Registros)
        if ~isempty(handles.Registros{i}.Plot)
            handles.Registros{i}.Plot.Visible = 'Off';
            handles.Registros{i}.TextPlot.Visible = 'Off';
        end
    end

    if ~isempty(handles.Registros{pos}.Plot)
        handles.Registros{pos}.Plot.Visible = 'On';
        handles.Registros{pos}.TextPlot.Visible = 'On';
    end

guidata(hObject,handles)

% --- Executes on button press in PB_Ubicar_Punto.
function PB_Ubicar_Punto_Callback(hObject, eventdata, handles)
```

```
    if strcmp(handles.TE_Registros.String,'0') return; end

    set(findall(handles.Panel_Registros,'Enable','On'),'Enable','Off');

    set(handles.Axes,'ButtonDownFcn',{@drawCurrPoint,handles.Tracking});
    set(handles.Imagen,'ButtonDownFcn',{@drawCurrPoint,handles.Tracking});

guidata(hObject, handles);

function drawCurrPoint(varargin)
    handles = guidata(varargin{3});
    pt = get(handles.Axes, 'CurrentPoint');
    x = pt(1,1);
    y = pt(1,2);
    axes(handles.Axes);

    pos = str2double(handles.TE_Registros.String);

    handles.Registros{pos}.X = x;
    handles.Registros{pos}.Y = y;

    if isempty(handles.Registros{pos}.Plot)
        handles.Registros{pos}.Plot = plot(x , y , 'LineStyle', 'none',...
            'Marker', 'o', 'Color',handles.PM_Color.BackgroundColor,...
            'MarkerFaceColor', handles.PM_Color.BackgroundColor, 'MarkerSize', 12);
        handles.Registros{pos}.TextPlot = text(x,y,num2str(pos),...
            'HorizontalAlignment', 'center', 'FontSize', 8, 'FontWeight', 'bold');
    else
        handles.Registros{pos}.Plot.XData = x;
        handles.Registros{pos}.Plot.YData = y;
        handles.Registros{pos}.TextPlot.Position = [x y];
    end

    set(handles.Axes,'ButtonDownFcn','');
    set(handles.Imagen,'ButtonDownFcn','');

    set(findall(handles.Panel_Registros,'Enable','Off'),'Enable','On');


guidata(varargin{3}, handles);

function TE_Texto_Registro_Callback(hObject, eventdata, handles)
    pos = str2double(handles.TE_Registros.String);
    handles.Registros{pos}.Texto = hObject.String;
guidata(hObject, handles);

% --- Executes on button press in PB_Eliminar_Registro.
function PB_Eliminar_Registro_Callback(hObject, eventdata, handles)

    pos = str2double(handles.TE_Registros.String);
    if ~pos return; end

    if ~isempty(handles.Registros{pos}.Plot)
        delete(handles.Registros{pos}.Plot);
        delete(handles.Registros{pos}.TextPlot);
    end

    handles.Registros(pos) = [];

    handles.SL_Registros.Value = ...
        min(handles.SL_Registros.Value, length(handles.Registros));

    handles.SL_Registros.Max = length(handles.Registros);
    if ~isempty(handles.Registros)
        handles.SL_Registros.SliderStep = [1 1].*1/(length(handles.Registros));
    else
        handles.SL_Registros.SliderStep = [1 1].*0;
    end

    for i = 1:length(handles.Registros)
        if ~isempty(handles.Registros{i}.Plot)
            handles.Registros{i}.TextPlot.String = num2str(i);
```

```
        end
    end

    SL_Registros_Callback(handles.SL_Registros, eventdata, handles);
    handles=guidata(hObject);

guidata(hObject,handles)

% --- Executes on button press in PB_Terminar_Edicion.
function PB_Terminar_Edicion_Callback(hObject, eventdata, handles)
    for i = 1:length(handles.Registros)
        if ~isempty(handles.Registros{i}.Plot)
            handles.Registros{i}.Plot.Visible = 'On';
            handles.Registros{i}.TextPlot.Visible = 'On';
        end
    end

    set(findall(handles.Panel_Registros,'Enable','On'),'Enable','Off');
    set(findall(handles.Panel_Menu_Trazo,'Enable','Of'),'Enable','On');
    set(findall(handles.Panel_Menu,'Enable','Of'),'Enable','On');
guidata(hObject, handles);


%Panel Menu
% --- Executes on button press in PB_Guardar.
function PB_Guardar_Callback(hObject, eventdata, handles)
% hObject    handle to PB_Guardar (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

    savefig(handles.output,[handles.path_proyecto '\Tracking\' ...
            handles.nombre_archivo '.fig']);

guidata(hObject,handles);



    % --- Executes on button press in PB_PDF.
function PB_PDF_Callback(hObject, eventdata, handles)

    TextDatos = [];

    TextDatos = [TextDatos 'Nombre del Proyecto: ' handles.nombre_proyecto '\n'];
    TextDatos = [TextDatos 'Nombre del Archivo: ' handles.nombre_archivo '\n\n'];
    TextDatos = [TextDatos 'Lugar de estudio: ' handles.lugar_estudio '\n'];
    TextDatos = [TextDatos 'Nombre del Usuario: ' handles.nombre_usuario '\n\n'];
    TextDatos = [TextDatos 'Condiciones climáticas: ' handles.Clima '\n'];
    TextDatos = [TextDatos 'Fecha: ' handles.TE_Dia.String '/' handles.TE_Mes.String
'/' handles.TE_Anio.String '\n'];
    TextDatos = [TextDatos 'Hora: ' handles.TE_Hora.String ':'
handles.TE_Minuto.String '\n\n'];


   % Crear la figura
    f = figure('Units', 'Pixels', 'Position', [200 30 210 300].*2);

    uicontrol(f,'Style','Text','Units', 'Pixels', 'Position', [10,280,190,10].*2,...
    'String', ['Tracking - ' handles.nombre_archivo], 'FontSize', 12,
'BackgroundColor','w');

    t = uicontrol(f,'Style','Text','Units', 'Pixels', 'Position',
[110,210,90,65].*2,...
    'FontSize', 6,  'HorizontalAlignment', 'Left',...
    'BackgroundColor','w');

    t.String = textwrap(t,{handles.Notas});

    t = uicontrol(f,'Style','Text','Units', 'Pixels', 'Position',
[10,210,90,65].*2,...
    'FontSize', 6,  'HorizontalAlignment', 'Left',...
    'BackgroundColor','w');
```

```
    t.String = textwrap(t,{sprintf(TextDatos)});

    h_axes = axes('Units', 'Pixels', 'Position', [10, 10, 190,190].*2);

    % Llenar axes
    imshow (handles.data_imagen); hold on;
    axis image;

    % Plot Figure

    plot(handles.Ruta.X, handles.Ruta.Y , 'Color', ...
        handles.PM_Color.BackgroundColor, 'LineWidth', 2);

    TextDatos = [];

    for i = 1:length(handles.Registros)
        R = handles.Registros{i};

        if ~isempty(R.Plot)
            plot(R.X , R.Y , 'LineStyle', 'none', 'Marker', 'o', 'Color', ...
                handles.PM_Color.BackgroundColor, 'MarkerFaceColor',...
                handles.PM_Color.BackgroundColor, 'MarkerSize', 12);
            text(R.X, R.Y, num2str(i), 'HorizontalAlignment', 'center',...
                'FontSize', 8, 'FontWeight', 'bold');
        end
        [r,~] = size(R.Texto);
        Texto = [];
        for j = 1: r
            Texto = [ Texto strtrim(R.Texto(j,:)) '\n'];
        end


        TextDatos = [TextDatos 'Registro ' num2str(i) ': ' Texto '\n'];

    end

    % Crear archivo
    pos = 1;
    while( exist([handles.path_proyecto '\Tracking\PDF\' handles.nombre_archivo '_'
num2str(pos) ' - Croquis.PDF'],'file'))
        pos = pos + 1;
    end

    figureName = [handles.path_proyecto '\Tracking\PDF\' handles.nombre_archivo '_'
num2str(pos) ' - Croquis.PDF'];
    set(gcf,'paperunits','centimeters');
    set(gcf,'papersize',[21 29.7]);
    set(gcf,'paperposition',[0,0,21,29.7]);
    print(gcf, '-dpdf',figureName);

    if ispc
        winopen(figureName);
    end

    close(f);
    % crear hoja con datos

    f = figure('Units', 'Pixels', 'Position', [200 30 210 300].*2);

    uicontrol(f,'Style','Text','Units', 'Pixels', 'Position', [10,280,190,10].*2,...
    'String', ['Tracking - ' handles.nombre_archivo], 'FontSize', 14,
'BackgroundColor','w');

    t = uicontrol(f,'Style','Text','Units', 'Pixels', 'Position',
[20,10,170,260].*2,...
    'FontSize', 6,  'HorizontalAlignment', 'Left',...
    'BackgroundColor','w');

    t.String = textwrap(t,{sprintf(TextDatos)});

    % Crear archivo
    pos = 1;
```

```
    while( exist([handles.path_proyecto '\Tracking\PDF\' handles.nombre_archivo '_'
num2str(pos) '.PDF'],'file'))
        pos = pos + 1;
    end

    figureName = [handles.path_proyecto '\Tracking\PDF\' handles.nombre_archivo '_'
num2str(pos) ' - Registros.PDF'];
    set(gcf,'paperunits','centimeters');
    set(gcf,'papersize',[21 29.7]);
    set(gcf,'paperposition',[0,0,21,29.7]);
    print(gcf, '-dpdf',figureName);
    if ispc
        winopen(figureName);
    end
close(f);


% --- Executes on button press in PB_Cerrar.
function PB_Cerrar_Callback(hObject, eventdata, handles)

    choice = questdlg(sprintf(['¿Desea guardar el archivo antes de cerrarlo?']), ...
                              'Cerrar archivo', ...
                              'Guardar','Continuar','Guardar');
    if strcmp(choice,'Guardar')
            PB_Guardar_Callback(handles.PB_Guardar,eventdata,handles);
    end

    openfig([handles.path_proyecto '\Pantalla_Principal.fig']);

delete(handles.output);
```

```
function varargout = Counting(varargin)
% COUNTING MATLAB code for Counting.fig
%      COUNTING, by itself, creates a new COUNTING or raises the existing
%      singleton*.
%
%      H = COUNTING returns the handle to a new COUNTING or the handle to
%      the existing singleton*.
%
%      COUNTING('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in COUNTING.M with the given input arguments.
%
%      COUNTING('Property','Value',...) creates a new COUNTING or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before Counting_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to Counting_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help Counting

% Last Modified by GUIDE v2.5 11-May-2016 19:32:43

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Counting_OpeningFcn, ...
                   'gui_OutputFcn',  @Counting_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

% --- Executes just before Counting is made visible.
function Counting_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to Counting (see VARARGIN)

warning('off','all')
set(findall(handles.Counting,'Units','Pixels'),'Units','Normalized');

handles.AppFolder = varargin{1}{1};
handles.path_proyecto = varargin{1}{2};
handles.nombre_proyecto = varargin{1}{3};
handles.nombre_archivo = varargin{1}{4};
handles.lugar_estudio = varargin{1}{5};
handles.nombre_usuario = varargin{1}{6};
handles.Counting.Name = handles.nombre_archivo;

handles.TE_Proyecto.String = handles.nombre_proyecto;
handles.TE_Archivo.String = handles.nombre_archivo;
handles.TE_Lugar.String = handles.lugar_estudio;
handles.Clima = '';
handles.Notas = 'Ingrese sus notas';
handles.allTable = cell(0,5);
handles.HistData = [];
```

```
handles.DurPeriodo = 10;
handles.CntPeriodo = 6;

handles.Periodos_elegidos = [1:handles.CntPeriodo];

handles.Seccion.Plot= [];
handles.Seccion.X= [];
handles.Seccion.Y = [];

handles.Visu.Sexo = 0;
handles.Visu.Edad = 0;
handles.Visu.Disc = 0;

set(findall(handles.Panel_Visu,'Enable','On'),'Enable','Off');
set(findall(handles.Panel_Axes,'Enable','On'),'Enable','Off');

handles.PB_Cerrar.Enable = 'On';

% Choose default command line output for Counting
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes Counting wait for user response (see UIRESUME)
% uiwait(handles.Counting);

% --- Outputs from this function are returned to the command line.
function varargout = Counting_OutputFcn(hObject, eventdata, handles)

% Get default command line output from handles structure
varargout{1} = handles.output;

%--------------------------------------------------------------------------
% Panel Informacion
% --- Executes on slider movement.
function SL_Hora_Callback(hObject, eventdata, handles)

    valor = round(hObject.Value);
    handles.TE_Hora.String = num2str(valor,'%02u');

guidata(hObject, handles);

% --- Executes on slider movement.
function SL_Minuto_Callback(hObject, eventdata, handles)

    valor = round(hObject.Value);
    handles.TE_Minuto.String = num2str(valor,'%02u');

guidata(hObject, handles);

% --- Executes on slider movement.
function SL_Dia_Callback(hObject, eventdata, handles)

    valor = round(hObject.Value);
    handles.TE_Dia.String = num2str(valor,'%02u');

guidata(hObject, handles);

% --- Executes on slider movement.
function SL_Mes_Callback(hObject, eventdata, handles)

    valor = round(hObject.Value);
    handles.TE_Mes.String = num2str(valor,'%02u');

guidata(hObject, handles);

% --- Executes on slider movement.
function SL_Anio_Callback(hObject, eventdata, handles)

    valor = round(hObject.Value);
```

```
    handles.TE_Anio.String = num2str(valor,'%04u');

guidata(hObject, handles);

function TE_Clima_Callback(hObject, eventdata, handles)

    handles.Clima = hObject.String;

guidata(hObject,handles)

% --- Executes on button press in PB_Notas.
function PB_Notas_Callback(hObject, eventdata, handles)

    answer = inputdlg({'Ingrese las notas - No exceda el espacio en la ventana'},...
                      'Notas - Counting', [12 50], {handles.Notas});
    if (isempty(answer)) return; end
    handles.Notas = answer{:};

guidata(hObject,handles);

% --- Executes on button press in PB_Importar_Imagen.
function PB_Importar_Imagen_Callback(hObject, eventdata, handles)

    image_path = uipickfiles('FilterSpec', [handles.AppFolder...
                 '\Proyectos\' handles.nombre_proyecto '\Imagenes\*.jpg'],...
                 'Prompt', 'Seleccione un archivo JPG', 'NumFiles', 1);

    if (~iscell(image_path) && ~image_path) return; end

    cla(handles.Axes)

    if (isempty(strfind(image_path{1},[handles.AppFolder...
        '\Proyectos\' handles.nombre_proyecto '\Imagenes\'])))

        copyfile(image_path{1},[handles.AppFolder '\Proyectos\'...
                 handles.nombre_proyecto '\Imagenes\Counting_'...
                 handles.nombre_archivo '.jpg']);
        image_path{1} = [handles.AppFolder '\Proyectos\'...
                 handles.nombre_proyecto '\Imagenes\Counting_'...
                 handles.nombre_archivo '.jpg'];
    end
    handles.data_imagen = imread( image_path{1} );
    axes(handles.Axes);
    handles.Imagen = imshow (handles.data_imagen); hold on;
    axis image;

    set(findall(handles.Panel_Axes,'Enable','Off'),'Enable','On');
    set(findall(handles.Panel_Visu,'Enable','Off'),'Enable','On');
    hObject.Enable = 'Off';

guidata(hObject,handles);

% Panel Axes
function TE_Duracion_Periodo_Callback(hObject, eventdata, handles)
    value = round(str2double(hObject.String),0);
    if isnan(value)
        msgbox('Ingrese un número entero positivo', 'Error');

    else
        handles.DurPeriodo = value;
    end
    hObject.String = handles.DurPeriodo;

guidata(hObject, handles);

function TE_Cantidad_Periodo_Callback(hObject, eventdata, handles)
value = round(str2double(hObject.String),0);
    if isnan(value)
        msgbox('Ingrese un número entero positivo', 'Error');
    else
        handles.CntPeriodo = value;
    end
```

```
    hObject.String = handles.CntPeriodo;

guidata(hObject, handles);
% --- Executes on slider movement.
function SL_Hora_Inicio_Callback(hObject, eventdata, handles)
    valor = round(hObject.Value);
    handles.TE_Hora_Inicio.String = num2str(valor,'%02u');

guidata(hObject, handles);

% --- Executes on slider movement.
function SL_Minuto_Inicio_Callback(hObject, eventdata, handles)
    valor = round(hObject.Value);
    handles.TE_Minuto_Inicio.String = num2str(valor,'%02u');

guidata(hObject, handles);

% --- Executes on button press in PB_Cargar_Datos.
function PB_Cargar_Datos_Callback(hObject, eventdata, handles)

    handles.allTable = Importar_Excel_Counting({handles.allTable,...
        handles.AppFolder, handles.nombre_proyecto});
    guidata(hObject, handles);
    Actualizar_visualizacion(hObject);
    handles = guidata(hObject);
guidata(hObject, handles);

% --- Executes on button press in PB_Dibujar_Seccion.
function PB_Dibujar_Seccion_Callback(hObject, eventdata, handles)
    if ~isempty (handles.Seccion.Plot)
        delete(handles.Seccion.Plot)
    end

    set(findall(handles.Counting,'Enable','On'),'Enable','Off');

    set(handles.Axes,'ButtonDownFcn',{@startDragFcn,handles.Counting});
    set(handles.Imagen,'ButtonDownFcn',{@startDragFcn,handles.Counting});

guidata(hObject, handles);

function startDragFcn(varargin)
    handles = guidata(varargin{3});
    set( handles.Counting, 'WindowButtonMotionFcn', {@draggingFcn,handles.Counting}
);
    set( handles.Counting, 'WindowButtonUpFcn', {@stopDragFcn,handles.Counting});
    pt = get(handles.Axes, 'CurrentPoint');
    x = pt(1,1);
    y = pt(1,2);
    axes(handles.Axes);
    handles.Seccion.X = x;
    handles.Seccion.Y = y;

    handles.Seccion.Plot =  plot(handles.Seccion.X , handles.Seccion.Y ,...
             'Color','r', 'LineWidth', 3);

guidata(varargin{3}, handles);

function draggingFcn(varargin)
    handles = guidata(varargin{3});
    pt = get(handles.Axes, 'CurrentPoint');

    x = pt(1,1);
    y = pt(1,2);

    handles.Seccion.X = [handles.Seccion.X x];
    handles.Seccion.Y = [handles.Seccion.Y y];
    handles.Seccion.Plot.XData = handles.Seccion.X;
    handles.Seccion.Plot.YData = handles.Seccion.Y;

guidata(varargin{3}, handles);

function stopDragFcn(varargin)
```

```
    handles = guidata(varargin{3});
    set(handles.Counting, 'WindowButtonMotionFcn', '');
    set(handles.Counting, 'WindowButtonUpFcn', '');
    set(handles.Axes,'ButtonDownFcn','');
    set(handles.Imagen,'ButtonDownFcn','');
    set(findall(handles.Counting,'Enable','Off'),'Enable','On');

guidata(varargin{3}, handles);

%Panel Menu

function Actualizar_visualizacion(hObject)

    handles = guidata(hObject);
    axes(handles.Axes_Histograma);

    handles.HistData = [];
    handles.StatData = [];

    [H,~] = size(handles.allTable);
    S = handles.Visu.Sexo;
    E = handles.Visu.Edad;
    D = handles.Visu.Disc;
    TS = {'','Masculino', 'Femenino'};
    TE = {'','05 - 12 años', '12 - 18 años', '18 - 60 años', '60 - 70 años',...
          '70 - 80 años', '80 - 90 años'};
    TD = {'','Ninguna Discapacidad', 'Discapacidad de Movimiento', 'Discapacidad
cognitiva'};

    for i = 1:H
        VSexo = find(strcmp(handles.allTable{i,3},TS));
        VEdad = find(strcmp(handles.allTable{i,4},TE));
        VDisc = find(strcmp(handles.allTable{i,5},TD));

        if isempty(VSexo) VSexo = 1; end
        if isempty(VEdad) VEdad = 1; end
        if isempty(VDisc) VDisc = 1; end

        if ( any(handles.Periodos_elegidos == handles.allTable{i,2}) &&...
             ( ~S || S==6   ||  bitand(S,bitsll(1,VSexo-1)) ) && ...
             ( ~E || E==126 ||  bitand(E,bitsll(1,VEdad-1)) ) && ...
             ( ~D || D==14  ||  bitand(D,bitsll(1,VDisc-1)) ) )
                handles.HistData = [ handles.HistData ,...
                ones(1,handles.allTable{i,1}) .* handles.allTable{i,2}];
        end
        handles.StatData = [ handles.StatData ,...
                ones(1,handles.allTable{i,1}) .* handles.allTable{i,2}];
    end

    hh = histogram (handles.Axes_Histograma,handles.StatData,...
                (0.75:.5:(handles.CntPeriodo+0.25)));

    StatData = hh.Values;

    StatData = StatData(StatData~=0);

    stats{1,1} =    sum(StatData); % Number of rows
    stats{2,1} =    min(StatData);
    stats{3,1} =    max(StatData);
    stats{4,1} = median(StatData);
    stats{5,1} =   mode(StatData);
    stats{6,1} =   mean(StatData);
    stats{7,1} =    std(StatData);

    hh = histogram (handles.Axes_Histograma,handles.HistData,...
                (0.75:.5:(handles.CntPeriodo+0.25)));

    StatData = hh.Values;

    StatData = StatData(StatData~=0);

    stats{1,2} =    sum(StatData); % Number of rows
    stats{2,2} =    min(StatData);
```

```
    stats{3,2} =    max(StatData);
    stats{4,2} = median(StatData);
    stats{5,2} =   mode(StatData);
    stats{6,2} =   mean(StatData);
    stats{7,2} =    std(StatData);
    handles.Tabla_Estadisticas.Data = stats;
    vtime = str2double(handles.TE_Hora_Inicio.String)*60 +...
    str2double(handles.TE_Minuto_Inicio.String);
    for i=1:handles.CntPeriodo
        Periodos{i} = [min2hour(vtime) ' - ' min2hour(vtime + handles.DurPeriodo)];
        vtime = vtime + handles.DurPeriodo;
    end
    if isempty(handles.HistData) return; end
    handles.Axes_Histograma.XLim = [min(handles.HistData)-
0.5,max(handles.HistData)+0.5];
    handles.Axes_Histograma.YLim = [0,(stats{3,2} * 1.1 )];
    handles.Axes_Histograma.XTick = 1:handles.CntPeriodo;
    handles.Axes_Histograma.XTickLabel = Periodos;
    handles.Axes_Histograma.TickLength = [0 0];
    handles.Axes_Histograma.XTickLabelRotation = 45;


    a = get(handles.Axes_Histograma,'XTickLabel');
    set(gca,'XTickLabel',a,'fontsize',8)

    n = hh.Values(1:2:(2*handles.CntPeriodo));
    x = 1:handles.CntPeriodo;
    barstrings = num2str(n');
    text(x,n,barstrings,'horizontalalignment','center','verticalalignment','bottom');



guidata (hObject,handles);

% --- Executes on button press in PB_Guardar.
function PB_Guardar_Callback(hObject, eventdata, handles)
% hObject    handle to PB_Guardar (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

    savefig(handles.output,[handles.path_proyecto '\Counting\' ...
            handles.nombre_archivo '.fig']);

guidata(hObject,handles);

% --- Executes on button press in RB_Sexo1.
function RB_Sexo1_Callback(hObject, eventdata, handles)
    handles.Visu.Sexo = bitset(handles.Visu.Sexo,2,hObject.Value);
    guidata(hObject,handles);
    Actualizar_visualizacion(hObject);
    handles = guidata(hObject);

guidata(hObject,handles);

% --- Executes on button press in RB_Sexo2.
function RB_Sexo2_Callback(hObject, eventdata, handles)
    handles.Visu.Sexo = bitset(handles.Visu.Sexo,3,hObject.Value);
    guidata(hObject,handles);
    Actualizar_visualizacion(hObject);
    handles = guidata(hObject);

guidata(hObject,handles);

% --- Executes on button press in RB_Disc1.
function RB_Disc1_Callback(hObject, eventdata, handles)
    handles.Visu.Disc = bitset(handles.Visu.Disc,2,hObject.Value);
    guidata(hObject,handles);
    Actualizar_visualizacion(hObject);
    handles = guidata(hObject);
guidata(hObject,handles);

% --- Executes on button press in RB_Disc2.
```

```
function RB_Disc2_Callback(hObject, eventdata, handles)
    handles.Visu.Disc = bitset(handles.Visu.Disc,3,hObject.Value);
    guidata(hObject,handles);
    Actualizar_visualizacion(hObject);
    handles = guidata(hObject);
guidata(hObject,handles);

% --- Executes on button press in RB_Disc3.
function RB_Disc3_Callback(hObject, eventdata, handles)
    handles.Visu.Disc = bitset(handles.Visu.Disc,4,hObject.Value);
    guidata(hObject,handles);
    Actualizar_visualizacion(hObject);
    handles = guidata(hObject);
guidata(hObject,handles);

% --- Executes on button press in RB_Edad1.
function RB_Edad1_Callback(hObject, eventdata, handles)
    handles.Visu.Edad = bitset(handles.Visu.Edad,2,hObject.Value);
    guidata(hObject,handles);
    Actualizar_visualizacion(hObject);
    handles = guidata(hObject);
guidata(hObject,handles);

% --- Executes on button press in RB_Edad2.
function RB_Edad2_Callback(hObject, eventdata, handles)
    handles.Visu.Edad = bitset(handles.Visu.Edad,3,hObject.Value);
    guidata(hObject,handles);
    Actualizar_visualizacion(hObject);
    handles = guidata(hObject);
guidata(hObject,handles);

% --- Executes on button press in RB_Edad3.
function RB_Edad3_Callback(hObject, eventdata, handles)
    handles.Visu.Edad = bitset(handles.Visu.Edad,4,hObject.Value);
    guidata(hObject,handles);
    Actualizar_visualizacion(hObject);
    handles = guidata(hObject);
guidata(hObject,handles);

% --- Executes on button press in RB_Edad4.
function RB_Edad4_Callback(hObject, eventdata, handles)
    handles.Visu.Edad = bitset(handles.Visu.Edad,5,hObject.Value);
    guidata(hObject,handles);
    Actualizar_visualizacion(hObject);
    handles = guidata(hObject);
guidata(hObject,handles);

% --- Executes on button press in RB_Edad5.
function RB_Edad5_Callback(hObject, eventdata, handles)
    handles.Visu.Edad = bitset(handles.Visu.Edad,6,hObject.Value);
    guidata(hObject,handles);
    Actualizar_visualizacion(hObject);
    handles = guidata(hObject);
guidata(hObject,handles);

% --- Executes on button press in RB_Edad6.
function RB_Edad6_Callback(hObject, eventdata, handles)
    handles.Visu.Edad = bitset(handles.Visu.Edad,7,hObject.Value);
    guidata(hObject,handles);
    Actualizar_visualizacion(hObject);
    handles = guidata(hObject);
guidata(hObject,handles);

% --- Executes on button press in PB_Seleccion_Periodo.
function PB_Seleccion_Periodo_Callback(hObject, eventdata, handles)

    Periodos = cell(1,handles.CntPeriodo);

    vtime = str2double(handles.TE_Hora_Inicio.String)*60 +...
            str2double(handles.TE_Minuto_Inicio.String);
    for i=1:handles.CntPeriodo
        Periodos{i} = [num2str(i) ' - ' min2hour(vtime) ' - ' ...
                       min2hour(vtime + handles.DurPeriodo)];
        vtime = vtime + handles.DurPeriodo;
```

```
    end

    [Selection,ok] = listdlg('PromptString','Seleccione los periodos que desea
visualizar',...
                     'ListString', Periodos, 'InitialValue',
handles.Periodos_elegidos,...
                     'SelectionMode', 'Multiple');
    if (~ok) return; end

    handles.Periodos_elegidos = Selection;
    guidata(hObject,handles)
    Actualizar_visualizacion(hObject);
    handles = guidata(hObject);
guidata(hObject,handles)


function strhora = min2hour(minutos)
    minutos = mod(minutos,60*24);
    strhora = [num2str(floor(minutos/60),'%02u') ':'
num2str(mod(minutos,60),'%02u')];


    % --- Executes on button press in PB_PDF.
function PB_PDF_Callback(hObject, eventdata, handles)

    TextDatos = [];

    TextDatos = [TextDatos 'Nombre del Proyecto: ' handles.nombre_proyecto '\n'];
    TextDatos = [TextDatos 'Nombre del Archivo: ' handles.nombre_archivo '\n\n'];
    TextDatos = [TextDatos 'Lugar de estudio: ' handles.lugar_estudio '\n'];
    TextDatos = [TextDatos 'Nombre del Usuario: ' handles.nombre_usuario '\n\n'];
    TextDatos = [TextDatos 'Condiciones climáticas: ' handles.Clima '\n'];
    TextDatos = [TextDatos 'Fecha: ' handles.TE_Dia.String '/' handles.TE_Mes.String
'/' handles.TE_Anio.String '\n'];
    TextDatos = [TextDatos 'Hora: ' handles.TE_Hora.String ':'
handles.TE_Minuto.String '\n\n'];

    TextDatos = [TextDatos 'Filtros aplicados:'];
    TextDatos = [TextDatos '\nEdad:'];
    if (handles.RB_Edad1.Value) TextDatos = [TextDatos ' ' handles.RB_Edad1.String];
end
    if (handles.RB_Edad2.Value) TextDatos = [TextDatos ' ' handles.RB_Edad2.String];
end
    if (handles.RB_Edad3.Value) TextDatos = [TextDatos ' ' handles.RB_Edad3.String];
end
    if (handles.RB_Edad4.Value) TextDatos = [TextDatos ' ' handles.RB_Edad4.String];
end
    if (handles.RB_Edad5.Value) TextDatos = [TextDatos ' ' handles.RB_Edad5.String];
end
    if (handles.RB_Edad6.Value) TextDatos = [TextDatos ' ' handles.RB_Edad6.String];
end
    if (TextDatos(end) == ':')  TextDatos = [TextDatos ' Ninguno']; end
    TextDatos = [TextDatos '\nSexo:'];
    if (handles.RB_Sexo1.Value) TextDatos = [TextDatos ' ' handles.RB_Sexo1.String];
end
    if (handles.RB_Sexo2.Value) TextDatos = [TextDatos ' ' handles.RB_Sexo2.String];
end
    if (TextDatos(end) == ':')  TextDatos = [TextDatos ' Ninguno']; end
    TextDatos = [TextDatos '\nDiscapacidad:'];
    if (handles.RB_Disc1.Value) TextDatos = [TextDatos ' ' handles.RB_Disc1.String];
end
    if (handles.RB_Disc2.Value) TextDatos = [TextDatos ' ' handles.RB_Disc2.String];
end
    if (handles.RB_Disc3.Value) TextDatos = [TextDatos ' ' handles.RB_Disc3.String];
end
    if (TextDatos(end) == ':')  TextDatos = [TextDatos ' Ninguno']; end

   % Crear la figura
    f = figure('Units', 'Pixels', 'Position', [200 30 210 300].*2);

    uicontrol(f,'Style','Text','Units', 'Pixels', 'Position', [10,280,190,10].*2,...
    'String', ['Counting - ' handles.nombre_archivo], 'FontSize', 12,
'BackgroundColor','w');
```

```
    t = uicontrol(f,'Style','Text','Units', 'Pixels', 'Position',
[110,210,90,65].*2,...
    'FontSize', 6,  'HorizontalAlignment', 'Left',...
    'BackgroundColor','w');

    t.String = textwrap(t,{handles.Notas});

    t = uicontrol(f,'Style','Text','Units', 'Pixels', 'Position',
[10,210,90,65].*2,...
    'FontSize', 6,  'HorizontalAlignment', 'Left',...
    'BackgroundColor','w');

    t.String = textwrap(t,{sprintf(TextDatos)});

    % Llenar axes Croquis

    h_axes = axes('Units', 'Pixels', 'Position', [10, 130, 80, 60].*2);

    imshow (handles.data_imagen); hold on;
    axis image;

    plot(handles.Seccion.X , handles.Seccion.Y , 'Color','r', 'LineWidth', 1.5);

    % tabla de estadisticas

    uitable(f, 'Position', [100 130 98 60].*2, 'ColumnWidth', {75 75},...
        'ColumnName', handles.Tabla_Estadisticas.ColumnName, 'RowName',...
        handles.Tabla_Estadisticas.RowName, 'Data',
handles.Tabla_Estadisticas.Data,...
        'BackgroundColor', handles.Tabla_Estadisticas.BackgroundColor, 'FontSize',
7);


    % Llenar histograma

    h_hist = axes('Units', 'Pixels', 'Position', [20, 30, 180, 80].*2, 'FontSize',6);

    hh = histogram (h_hist,handles.HistData,(0.75:.5:(handles.CntPeriodo+0.25)));

    vtime = str2double(handles.TE_Hora_Inicio.String)*60 +...
    str2double(handles.TE_Minuto_Inicio.String);

    for i=1:handles.CntPeriodo
        Periodos{i} = [min2hour(vtime) ' - ' min2hour(vtime + handles.DurPeriodo)];
        vtime = vtime + handles.DurPeriodo;
    end

    h_hist.XLim = [min(handles.HistData)-0.5,max(handles.HistData)+0.5];
    h_hist.YLim = [0, (handles.Tabla_Estadisticas.Data{3,2} * 1.1)];
    h_hist.XTick = 1:handles.CntPeriodo;
    h_hist.XTickLabel = Periodos;
    h_hist.TickLength = [0 0];
    h_hist.XTickLabelRotation = 45;

    a = get(h_hist,'XTickLabel');
    set(gca,'XTickLabel',a,'fontsize',6)

    n = hh.Values(1:2:(2*handles.CntPeriodo));
    x = 1:handles.CntPeriodo;
    barstrings = num2str(n');

text(x,n,barstrings,'horizontalalignment','center','verticalalignment','bottom','Font
Size',6)

    % Crear archivo
    pos = 1;
    while( exist([handles.path_proyecto '\Counting\PDF\' handles.nombre_archivo '_'
num2str(pos) '.PDF'],'file'))
        pos = pos + 1;
    end
```

```
    figureName = [handles.path_proyecto '\Counting\PDF\' handles.nombre_archivo '_'
num2str(pos) '.PDF'];
    set(gcf,'paperunits','centimeters');
    set(gcf,'papersize',[21 29.7]);
    set(gcf,'paperposition',[0,0,21,29.7]);
    print(gcf, '-dpdf',figureName);
    if ispc
        winopen(figureName);
    end
close(f);

% --- Executes on button press in PB_Cerrar.
function PB_Cerrar_Callback(hObject, eventdata, handles)

    choice = questdlg(sprintf(['¿Desea guardar el archivo antes de cerrarlo?']), ...
                              'Cerrar archivo', ...
                              'Guardar','Continuar','Guardar');
    if strcmp(choice,'Guardar')
            PB_Guardar_Callback(handles.PB_Guardar,eventdata,handles);
    end

    openfig([handles.path_proyecto '\Pantalla_Principal.fig']);

delete(handles.output);
```

```
function varargout = Speed_Test(varargin)
% SPEED_TEST MATLAB code for Speed_Test.fig
%      SPEED_TEST, by itself, creates a new SPEED_TEST or raises the existing
%      singleton*.
%
%      H = SPEED_TEST returns the handle to a new SPEED_TEST or the handle to
%      the existing singleton*.
%
%      SPEED_TEST('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in SPEED_TEST.M with the given input arguments.
%
%      SPEED_TEST('Property','Value',...) creates a new SPEED_TEST or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before Speed_Test_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to Speed_Test_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help Speed_Test

% Last Modified by GUIDE v2.5 17-May-2016 20:56:28

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Speed_Test_OpeningFcn, ...
                   'gui_OutputFcn',  @Speed_Test_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

% --- Executes just before Speed_Test is made visible.
function Speed_Test_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to Speed_Test (see VARARGIN)

warning('off','all')
set(findall(handles.Speed_Test,'Units','Pixels'),'Units','Normalized');

handles.AppFolder = varargin{1}{1};
handles.path_proyecto = varargin{1}{2};
handles.nombre_proyecto = varargin{1}{3};
handles.nombre_archivo = varargin{1}{4};
handles.lugar_estudio = varargin{1}{5};
handles.nombre_usuario = varargin{1}{6};
handles.Speed_Test.Name = handles.nombre_archivo;

handles.TE_Proyecto.String = handles.nombre_proyecto;
handles.TE_Archivo.String = handles.nombre_archivo;
handles.TE_Lugar.String = handles.lugar_estudio;
handles.Clima = '';
handles.Notas = 'Ingrese sus notas';

handles.allTable = cell(0,4);
handles.HistData = [];
```

```
handles.Distancia = [];

handles.PM_Unidad.String = {'m/s', 'km/h'};

handles.Seccion.Plot= cell(1,2);
handles.Seccion.X = cell(1,2);
handles.Seccion.Y = cell(1,2);

handles.Visu.Sexo = 0;
handles.Visu.Edad = 0;
handles.Visu.Disc = 0;

set(findall(handles.Panel_Visu,'Enable','On'),'Enable','Off');
set(findall(handles.Panel_Axes,'Enable','On'),'Enable','Off');

handles.PB_Cerrar.Enable = 'On';

% Choose default command line output for Speed_Test
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes Speed_Test wait for user response (see UIRESUME)
% uiwait(handles.Speed_Test);

% --- Outputs from this function are returned to the command line.
function varargout = Speed_Test_OutputFcn(hObject, eventdata, handles)

% Get default command line output from handles structure
varargout{1} = handles.output;

%--------------------------------------------------------------------------
% Panel Informacion
% --- Executes on slider movement.
function SL_Hora_Callback(hObject, eventdata, handles)

    valor = round(hObject.Value);
    handles.TE_Hora.String = num2str(valor,'%02u');

guidata(hObject, handles);

% --- Executes on slider movement.
function SL_Minuto_Callback(hObject, eventdata, handles)

    valor = round(hObject.Value);
    handles.TE_Minuto.String = num2str(valor,'%02u');

guidata(hObject, handles);

% --- Executes on slider movement.
function SL_Dia_Callback(hObject, eventdata, handles)

    valor = round(hObject.Value);
    handles.TE_Dia.String = num2str(valor,'%02u');

guidata(hObject, handles);

% --- Executes on slider movement.
function SL_Mes_Callback(hObject, eventdata, handles)

    valor = round(hObject.Value);
    handles.TE_Mes.String = num2str(valor,'%02u');

guidata(hObject, handles);

% --- Executes on slider movement.
function SL_Anio_Callback(hObject, eventdata, handles)

    valor = round(hObject.Value);
```

```
    handles.TE_Anio.String = num2str(valor,'%04u');

guidata(hObject, handles);

function TE_Clima_Callback(hObject, eventdata, handles)

    handles.Clima = hObject.String;

guidata(hObject,handles)

% --- Executes on button press in PB_Notas.
function PB_Notas_Callback(hObject, eventdata, handles)

    answer = inputdlg({'Ingrese las notas - No exceda el espacio en la ventana'},...
                      'Notas - Speed Test', [12 50], {handles.Notas});
    if (isempty(answer)) return; end
    handles.Notas = answer{:};

guidata(hObject,handles);

% --- Executes on button press in PB_Importar_Imagen.
function PB_Importar_Imagen_Callback(hObject, eventdata, handles)

    image_path = uipickfiles('FilterSpec', [handles.AppFolder...
                 '\Proyectos\' handles.nombre_proyecto '\Imagenes\*.jpg'],...
                 'Prompt', 'Seleccione un archivo JPG', 'NumFiles', 1);

    if (~iscell(image_path) && ~image_path) return; end

    if (isempty(strfind(image_path{1},[handles.AppFolder...
        '\Proyectos\' handles.nombre_proyecto '\Imagenes\'])))

        cla(handles.Axes)

        copyfile(image_path{1},[handles.AppFolder '\Proyectos\'...
                 handles.nombre_proyecto '\Imagenes\Counting_'...
                 handles.nombre_archivo '.jpg']);
        image_path{1} = [handles.AppFolder '\Proyectos\'...
                 handles.nombre_proyecto '\Imagenes\Counting_'...
                 handles.nombre_archivo '.jpg'];
    end
    handles.data_imagen = imread( image_path{1} );
    axes(handles.Axes);
    handles.Imagen = imshow (handles.data_imagen); hold on;
    axis image;

    set(findall(handles.Panel_Axes,'Enable','Off'),'Enable','On');
    set(findall(handles.Panel_Visu,'Enable','Off'),'Enable','On');
    hObject.Enable = 'Off';

guidata(hObject,handles);

% Panel Axes
% --- Executes on button press in PB_Cargar_Datos.
function PB_Cargar_Datos_Callback(hObject, eventdata, handles)

    [handles.allTable,handles.Distancia] = ...
        Importar_Excel_Speed_Test({handles.allTable,handles.AppFolder,...
        handles.Distancia, handles.nombre_proyecto});

    guidata(hObject, handles);
    Actualizar_visualizacion(hObject);
    handles = guidata(hObject);

guidata(hObject, handles);

% --- Executes on button press in PB_Dibujar_Seccion.
function PB_Dibujar_Seccion_Callback(hObject, eventdata, handles)
    if ~isempty (handles.Seccion.Plot{1})
        delete(handles.Seccion.Plot{1});
        delete(handles.Seccion.Plot{2});
    end
```

```
    handles.Cnt_trazos = 0;

    set(findall(handles.Speed_Test,'Enable','On'),'Enable','Off');

    set(handles.Axes,'ButtonDownFcn',{@startDragFcn,handles.Speed_Test});
    set(handles.Imagen,'ButtonDownFcn',{@startDragFcn,handles.Speed_Test});

guidata(hObject, handles);

function startDragFcn(varargin)
    handles = guidata(varargin{3});
    handles.Cnt_trazos = handles.Cnt_trazos +1;
    set( handles.Speed_Test, 'WindowButtonMotionFcn',
{@draggingFcn,handles.Speed_Test} );
    set( handles.Speed_Test, 'WindowButtonUpFcn', {@stopDragFcn,handles.Speed_Test});
    pt = get(handles.Axes, 'CurrentPoint');
    x = pt(1,1);
    y = pt(1,2);
    axes(handles.Axes);
    handles.Seccion.X{handles.Cnt_trazos} = x;
    handles.Seccion.Y{handles.Cnt_trazos} = y;

    handles.Seccion.Plot{handles.Cnt_trazos} =
plot(handles.Seccion.X{handles.Cnt_trazos},...
                        handles.Seccion.Y{handles.Cnt_trazos}, 'Color','r',
'LineWidth', 3);

guidata(varargin{3}, handles);

function draggingFcn(varargin)
    handles = guidata(varargin{3});
    pt = get(handles.Axes, 'CurrentPoint');

    x = pt(1,1);
    y = pt(1,2);

    handles.Seccion.X{handles.Cnt_trazos} = [handles.Seccion.X{handles.Cnt_trazos}
x];
    handles.Seccion.Y{handles.Cnt_trazos} = [handles.Seccion.Y{handles.Cnt_trazos}
y];
    handles.Seccion.Plot{handles.Cnt_trazos}.XData =
handles.Seccion.X{handles.Cnt_trazos};
    handles.Seccion.Plot{handles.Cnt_trazos}.YData =
handles.Seccion.Y{handles.Cnt_trazos};

guidata(varargin{3}, handles);

function stopDragFcn(varargin)
    handles = guidata(varargin{3});
    if (handles.Cnt_trazos == 2)
        set(handles.Axes,'ButtonDownFcn','');
        set(handles.Imagen,'ButtonDownFcn','');
        set(findall(handles.Speed_Test,'Enable','Off'),'Enable','On');
    end
    set(handles.Speed_Test, 'WindowButtonMotionFcn', '');
    set(handles.Speed_Test, 'WindowButtonUpFcn', '');

guidata(varargin{3}, handles);

function Actualizar_visualizacion(hObject)

    handles = guidata(hObject);
    axes(handles.Axes_Histograma);

    handles.HistData = [];
    handles.StatData = [];

    [H,~] = size(handles.allTable);
    S = handles.Visu.Sexo;
    E = handles.Visu.Edad;
    D = handles.Visu.Disc;
```

```
    TS = {'','Masculino', 'Femenino'};
    TE = {'','05 - 12 años', '12 - 18 años', '18 - 60 años', '60 - 70 años',...
          '70 - 80 años', '80 - 90 años'};
    TD = {'','Ninguna Discapacidad', 'Discapacidad de Movimiento', 'Discapacidad
cognitiva'};

    for i = 1:H
        VSexo = find(strcmp(handles.allTable{i,2},TS));
        VEdad = find(strcmp(handles.allTable{i,3},TE));
        VDisc = find(strcmp(handles.allTable{i,4},TD));
        if isempty(VSexo) VSexo = 1; end
        if isempty(VEdad) VEdad = 1; end
        if isempty(VDisc) VDisc = 1; end
        if ( ( ~S || S==6   ||  bitand(S,bitsll(1,VSexo-1)) ) && ...
             ( ~E || E==126 ||  bitand(E,bitsll(1,VEdad-1)) ) && ...
             ( ~D || D==14  ||  bitand(D,bitsll(1,VDisc-1)) ) )
                handles.HistData = [ handles.HistData ,
handles.Distancia/handles.allTable{i,1} ];
        end
        handles.StatData = [ handles.StatData ,
handles.Distancia/handles.allTable{i,1} ];
    end

    strLabelX = 'Velocidad (m/s)';
    if strcmp(handles.PM_Unidad.String{handles.PM_Unidad.Value},'km/h')
        handles.HistData = handles.HistData.*(18/5);
        handles.StatData = handles.StatData.*(18/5);
        strLabelX = 'Velocidad (km/h)';
    end



    stats{1,1} = length(handles.StatData);
    stats{2,1} =    min(handles.StatData);
    stats{3,1} =    max(handles.StatData);
    stats{4,1} = median(handles.StatData);
    stats{5,1} =   mode(handles.StatData);
    stats{6,1} =   mean(handles.StatData);
    stats{7,1} =    std(handles.StatData);

    hh = histogram (handles.Axes_Histograma,handles.HistData);

    stats{1,2} = length(handles.HistData);
    stats{2,2} =    min(handles.HistData);
    stats{3,2} =    max(handles.HistData);
    stats{4,2} = median(handles.HistData);
    stats{5,2} =   mode(handles.HistData);
    stats{6,2} =   mean(handles.HistData);
    stats{7,2} =    std(handles.HistData);
    handles.Tabla_Estadisticas.Data = stats;
    if isempty(handles.HistData) return; end
    xlabel(strLabelX);
    handles.Axes_Histograma.YLim = [0,(max(hh.Values) * 1.1 )];

    a = get(handles.Axes_Histograma,'XTickLabel');
    set(gca,'XTickLabel',a,'fontsize',8)

    n = hh.Values;
    x = ones(1,hh.NumBins);
    for i = 1:hh.NumBins
        x(i) = (hh.BinEdges(i) + hh.BinEdges(i+1))/2;
    end
    barstrings = num2str(n');
    text(x,n,barstrings,'horizontalalignment','center','verticalalignment','bottom')



guidata (hObject,handles);

% --- Executes on button press in PB_Guardar.
function PB_Guardar_Callback(hObject, eventdata, handles)
% hObject    handle to PB_Guardar (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

    savefig(handles.output,[handles.path_proyecto '\Speed Test\' ...
            handles.nombre_archivo '.fig']);

guidata(hObject,handles);

% --- Executes on selection change in PM_Unidad.
function PM_Unidad_Callback(hObject, eventdata, handles)
    Actualizar_visualizacion(hObject);
    handles = guidata(hObject);
guidata(hObject,handles);

% --- Executes on button press in RB_Sexo1.
function RB_Sexo1_Callback(hObject, eventdata, handles)
    handles.Visu.Sexo = bitset(handles.Visu.Sexo,2,hObject.Value);
    guidata(hObject,handles);
    Actualizar_visualizacion(hObject);
    handles = guidata(hObject);
guidata(hObject,handles);

% --- Executes on button press in RB_Sexo2.
function RB_Sexo2_Callback(hObject, eventdata, handles)
    handles.Visu.Sexo = bitset(handles.Visu.Sexo,3,hObject.Value);
    guidata(hObject,handles);
    Actualizar_visualizacion(hObject);
    handles = guidata(hObject);
guidata(hObject,handles);

% --- Executes on button press in RB_Disc1.
function RB_Disc1_Callback(hObject, eventdata, handles)
    handles.Visu.Disc = bitset(handles.Visu.Disc,2,hObject.Value);
    guidata(hObject,handles);
    Actualizar_visualizacion(hObject);
    handles = guidata(hObject);
guidata(hObject,handles);

% --- Executes on button press in RB_Disc2.
function RB_Disc2_Callback(hObject, eventdata, handles)
    handles.Visu.Disc = bitset(handles.Visu.Disc,3,hObject.Value);
    guidata(hObject,handles);
    Actualizar_visualizacion(hObject);
    handles = guidata(hObject);
guidata(hObject,handles);

% --- Executes on button press in RB_Disc3.
function RB_Disc3_Callback(hObject, eventdata, handles)
    handles.Visu.Disc = bitset(handles.Visu.Disc,4,hObject.Value);
    guidata(hObject,handles);
    Actualizar_visualizacion(hObject);
    handles = guidata(hObject);
guidata(hObject,handles);

% --- Executes on button press in RB_Edad1.
function RB_Edad1_Callback(hObject, eventdata, handles)
    handles.Visu.Edad = bitset(handles.Visu.Edad,2,hObject.Value);
    guidata(hObject,handles);
    Actualizar_visualizacion(hObject);
    handles = guidata(hObject);
guidata(hObject,handles);

% --- Executes on button press in RB_Edad2.
function RB_Edad2_Callback(hObject, eventdata, handles)
    handles.Visu.Edad = bitset(handles.Visu.Edad,3,hObject.Value);
    guidata(hObject,handles);
    Actualizar_visualizacion(hObject);
    handles = guidata(hObject);
guidata(hObject,handles);

% --- Executes on button press in RB_Edad3.
function RB_Edad3_Callback(hObject, eventdata, handles)
    handles.Visu.Edad = bitset(handles.Visu.Edad,4,hObject.Value);
    guidata(hObject,handles);
```

```
    Actualizar_visualizacion(hObject);
    handles = guidata(hObject);
guidata(hObject,handles);

% --- Executes on button press in RB_Edad4.
function RB_Edad4_Callback(hObject, eventdata, handles)
    handles.Visu.Edad = bitset(handles.Visu.Edad,5,hObject.Value);
    guidata(hObject,handles);
    Actualizar_visualizacion(hObject);
    handles = guidata(hObject);
guidata(hObject,handles);

% --- Executes on button press in RB_Edad5.
function RB_Edad5_Callback(hObject, eventdata, handles)
    handles.Visu.Edad = bitset(handles.Visu.Edad,6,hObject.Value);
    guidata(hObject,handles);
    Actualizar_visualizacion(hObject);
    handles = guidata(hObject);
guidata(hObject,handles);

% --- Executes on button press in RB_Edad6.
function RB_Edad6_Callback(hObject, eventdata, handles)
    handles.Visu.Edad = bitset(handles.Visu.Edad,7,hObject.Value);
    guidata(hObject,handles);
    Actualizar_visualizacion(hObject);
    handles = guidata(hObject);
guidata(hObject,handles);

    % --- Executes on button press in PB_PDF.
function PB_PDF_Callback(hObject, eventdata, handles)
TextDatos = [];

    TextDatos = [TextDatos 'Nombre del Proyecto: ' handles.nombre_proyecto '\n'];
    TextDatos = [TextDatos 'Nombre del Archivo: ' handles.nombre_archivo '\n\n'];
    TextDatos = [TextDatos 'Lugar de estudio: ' handles.lugar_estudio '\n'];
    TextDatos = [TextDatos 'Nombre del Usuario: ' handles.nombre_usuario '\n\n'];
    TextDatos = [TextDatos 'Condiciones climáticas: ' handles.Clima '\n'];
    TextDatos = [TextDatos 'Fecha: ' handles.TE_Dia.String '/' handles.TE_Mes.String
'/' handles.TE_Anio.String '\n'];
    TextDatos = [TextDatos 'Hora: ' handles.TE_Hora.String ':'
handles.TE_Minuto.String '\n\n'];

    TextDatos = [TextDatos 'Filtros aplicados:'];
    TextDatos = [TextDatos '\nEdad:'];
    if (handles.RB_Edad1.Value) TextDatos = [TextDatos ' ' handles.RB_Edad1.String];
end
    if (handles.RB_Edad2.Value) TextDatos = [TextDatos ' ' handles.RB_Edad2.String];
end
    if (handles.RB_Edad3.Value) TextDatos = [TextDatos ' ' handles.RB_Edad3.String];
end
    if (handles.RB_Edad4.Value) TextDatos = [TextDatos ' ' handles.RB_Edad4.String];
end
    if (handles.RB_Edad5.Value) TextDatos = [TextDatos ' ' handles.RB_Edad5.String];
end
    if (handles.RB_Edad6.Value) TextDatos = [TextDatos ' ' handles.RB_Edad6.String];
end
    if (TextDatos(end) == ':')  TextDatos = [TextDatos ' Ninguno']; end
    TextDatos = [TextDatos '\nSexo:'];
    if (handles.RB_Sexo1.Value) TextDatos = [TextDatos ' ' handles.RB_Sexo1.String];
end
    if (handles.RB_Sexo2.Value) TextDatos = [TextDatos ' ' handles.RB_Sexo2.String];
end
    if (TextDatos(end) == ':')  TextDatos = [TextDatos ' Ninguno']; end
    TextDatos = [TextDatos '\nDiscapacidad:'];
    if (handles.RB_Disc1.Value) TextDatos = [TextDatos ' ' handles.RB_Disc1.String];
end
    if (handles.RB_Disc2.Value) TextDatos = [TextDatos ' ' handles.RB_Disc2.String];
end
    if (handles.RB_Disc3.Value) TextDatos = [TextDatos ' ' handles.RB_Disc3.String];
end
    if (TextDatos(end) == ':')  TextDatos = [TextDatos ' Ninguno']; end

   % Crear la figura
    f = figure('Units', 'Pixels', 'Position', [200 30 210 300].*2);
```

```
    uicontrol(f,'Style','Text','Units', 'Pixels', 'Position', [10,280,190,10].*2,...
    'String', ['Speed Test - ' handles.nombre_archivo], 'FontSize', 12,
'BackgroundColor','w');

    t = uicontrol(f,'Style','Text','Units', 'Pixels', 'Position',
[110,210,90,65].*2,...
    'FontSize', 6,  'HorizontalAlignment', 'Left',...
    'BackgroundColor','w');

    t.String = textwrap(t,{handles.Notas});

    t = uicontrol(f,'Style','Text','Units', 'Pixels', 'Position',
[10,210,90,65].*2,...
    'FontSize', 6,  'HorizontalAlignment', 'Left',...
    'BackgroundColor','w');

    t.String = textwrap(t,{sprintf(TextDatos)});

    % Llenar axes Croquis

    h_axes = axes('Units', 'Pixels', 'Position', [10, 130, 80, 60].*2);

    imshow (handles.data_imagen); hold on;
    axis image;

    plot(handles.Seccion.X{1}, handles.Seccion.Y{1}, 'Color','r', 'LineWidth', 1.5);
    plot(handles.Seccion.X{2}, handles.Seccion.Y{2}, 'Color','r', 'LineWidth', 1.5);

    title(['Distancia = ' num2str(handles.Distancia) ' m']);

    % tabla de estadisticas

    uitable(f, 'Position', [100 130 98 60].*2, 'ColumnWidth', {75 75},...
        'ColumnName', handles.Tabla_Estadisticas.ColumnName, 'RowName',...
        handles.Tabla_Estadisticas.RowName, 'Data',
handles.Tabla_Estadisticas.Data,...
        'BackgroundColor', handles.Tabla_Estadisticas.BackgroundColor, 'FontSize',
7);


    % Llenar histograma

    h_hist = axes('Units', 'Pixels', 'Position', [20, 30, 180, 80].*2, 'FontSize',6);
    hh = histogram (h_hist,handles.HistData);

    xlabel(handles.Axes_Histograma.XLabel.String);
    h_hist.YLim = [0,(max(hh.Values) * 1.1 )];

    a = get(h_hist,'XTickLabel');
    set(gca,'XTickLabel',a,'fontsize',6)

    n = hh.Values;
    x = ones(1,hh.NumBins);
    for i = 1:hh.NumBins
        x(i) = (hh.BinEdges(i) + hh.BinEdges(i+1))/2;
    end
    barstrings = num2str(n');

text(x,n,barstrings,'horizontalalignment','center','verticalalignment','bottom','Font
Size',6)

    % Crear archivo
    pos = 1;
    while( exist([handles.path_proyecto '\Speed Test\PDF\' handles.nombre_archivo '_'
num2str(pos) '.PDF'],'file'))
        pos = pos + 1;
    end

    figureName = [handles.path_proyecto '\Speed Test\PDF\' handles.nombre_archivo '_'
num2str(pos) '.PDF'];
    set(gcf,'paperunits','centimeters');
    set(gcf,'papersize',[21 29.7]);
```

```
    set(gcf,'paperposition',[0,0,21,29.7]);
    print(gcf, '-dpdf',figureName);
    if ispc
        winopen(figureName);
    end
close(f);

% --- Executes on button press in PB_Cerrar.
function PB_Cerrar_Callback(hObject, eventdata, handles)

    choice = questdlg(sprintf(['¿Desea guardar el archivo antes de cerrarlo?']), ...
                              'Cerrar archivo', ...
                              'Guardar','Continuar','Guardar');
    if strcmp(choice,'Guardar')
            PB_Guardar_Callback(handles.PB_Guardar,eventdata,handles);
    end

    openfig([handles.path_proyecto '\Pantalla_Principal.fig']);

delete(handles.output);
```

```
function varargout = Diary(varargin)
% DIARY MATLAB code for Diary.fig
%      DIARY, by itself, creates a new DIARY or raises the existing
%      singleton*.
%
%      H = DIARY returns the handle to a new DIARY or the handle to
%      the existing singleton*.
%
%      DIARY('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in DIARY.M with the given input arguments.
%
%      DIARY('Property','Value',...) creates a new DIARY or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before Diary_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to Diary_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help Diary

% Last Modified by GUIDE v2.5 19-May-2016 16:52:28

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Diary_OpeningFcn, ...
                   'gui_OutputFcn',  @Diary_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

% --- Executes just before Diary is made visible.
function Diary_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to Diary (see VARARGIN)

warning('off','all')
set(findall(handles.Diary,'Units','Pixels'),'Units','Normalized');

handles.AppFolder = varargin{1}{1};
handles.path_proyecto = varargin{1}{2};
handles.nombre_proyecto = varargin{1}{3};
handles.nombre_archivo = varargin{1}{4};
handles.lugar_estudio = varargin{1}{5};
handles.nombre_usuario = varargin{1}{6};

handles.Registros = cell(0);

handles.Diary.Name = handles.nombre_archivo;

handles.TE_Proyecto.String = handles.nombre_proyecto;
handles.TE_Archivo.String = handles.nombre_archivo;
handles.TE_Lugar.String = handles.lugar_estudio;

handles.Clima = '';
```

```
set(findall(handles.Panel_Registros,'Enable','On'),'Enable','Off');
handles.PB_Nuevo_Registro.Enable = 'On';
handles.T_Registro.Enable = 'On';
handles.TE_Registros.Enable = 'On';
handles.SL_Registros.Enable = 'On';


% Choose default command line output for Diary
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes Diary wait for user response (see UIRESUME)
% uiwait(handles.Diary);

% --- Outputs from this function are returned to the command line.
function varargout = Diary_OutputFcn(hObject, eventdata, handles)

% Get default command line output from handles structure
varargout{1} = handles.output;

%--------------------------------------------------------------------------
% Panel Informacion
% --- Executes on slider movement.
function SL_Hora_Callback(hObject, eventdata, handles)

    valor = round(hObject.Value);
    handles.TE_Hora.String = num2str(valor,'%02u');

guidata(hObject, handles);

% --- Executes on slider movement.
function SL_Minuto_Callback(hObject, eventdata, handles)

    valor = round(hObject.Value);
    handles.TE_Minuto.String = num2str(valor,'%02u');

guidata(hObject, handles);

% --- Executes on slider movement.
function SL_Dia_Callback(hObject, eventdata, handles)

    valor = round(hObject.Value);
    handles.TE_Dia.String = num2str(valor,'%02u');

guidata(hObject, handles);

% --- Executes on slider movement.
function SL_Mes_Callback(hObject, eventdata, handles)

    valor = round(hObject.Value);
    handles.TE_Mes.String = num2str(valor,'%02u');

guidata(hObject, handles);

% --- Executes on slider movement.
function SL_Anio_Callback(hObject, eventdata, handles)

    valor = round(hObject.Value);
    handles.TE_Anio.String = num2str(valor,'%04u');

guidata(hObject, handles);

function TE_Clima_Callback(hObject, eventdata, handles)

    handles.Clima = hObject.String;

guidata(hObject,handles)
```

```
% --- Executes on button press in PB_Notas.
function PB_Notas_Callback(hObject, eventdata, handles)

    answer = inputdlg({'Ingrese las notas - No exceda el espacio en la ventana'},...
                      'Notas - Diary', [12 50], {handles.Notas});
    if (isempty(answer)) return; end
    handles.Notas = answer{:};

guidata(hObject,handles);

% --- Executes on slider movement.
function SL_Hora_Registro_Callback(hObject, eventdata, handles)
    valor = round(hObject.Value);
    handles.TE_Hora_Registro.String = num2str(valor,'%02u');
    pos = str2double(handles.TE_Registros.String);
    handles.Registros{pos}.Hora = num2str(valor,'%02u');

guidata(hObject, handles);

% --- Executes on slider movement.
function SL_Minuto_Registro_Callback(hObject, eventdata, handles)
    valor = round(hObject.Value);
    handles.TE_Minuto_Registro.String = num2str(valor,'%02u');
    pos = str2double(handles.TE_Registros.String);
    handles.Registros{pos}.Minuto = num2str(valor,'%02u');

guidata(hObject, handles);

% --- Executes on slider movement.
function SL_Dia_Registro_Callback(hObject, eventdata, handles)

    valor = round(hObject.Value);
    handles.TE_Dia_Registro.String = num2str(valor,'%02u');
    pos = str2double(handles.TE_Registros.String);
    handles.Registros{pos}.Dia = num2str(valor,'%02u');

guidata(hObject, handles);

% --- Executes on slider movement.
function SL_Mes_Registro_Callback(hObject, eventdata, handles)

    valor = round(hObject.Value);
    handles.TE_Mes_Registro.String = num2str(valor,'%02u');
    pos = str2double(handles.TE_Registros.String);
    handles.Registros{pos}.Mes = num2str(valor,'%02u');

guidata(hObject, handles);

% --- Executes on slider movement.
function SL_Anio_Registro_Callback(hObject, eventdata, handles)

    valor = round(hObject.Value);
    handles.TE_Anio_Registro.String = num2str(valor,'%04u');
    pos = str2double(handles.TE_Registros.String);
    handles.Registros{pos}.Anio = num2str(valor,'%04u');

guidata(hObject, handles);

%Panel Registros
% --- Executes on button press in PB_Nuevo_Registro.
function PB_Nuevo_Registro_Callback(hObject, eventdata, handles)

    N = length(handles.Registros);

    handles.Registros{N+1} = struct( 'X', [], 'Y', [], 'Plot', [],...
            'Texto', 'Registro', 'TextPlot', [], ...
            'data_imagen', [], 'Hora', '12', 'Minuto', '00', 'Dia', '01',...
            'Mes', '01', 'Anio','2016');

    handles.SL_Registros.Max = N+1;
    handles.SL_Registros.Value = N+1;
```

```
    handles.SL_Registros.SliderStep = [1 1].*1/(N+1);

    SL_Registros_Callback(handles.SL_Registros, eventdata, handles);
    handles=guidata(hObject);
guidata(hObject,handles)


% --- Executes on slider movement.
function SL_Registros_Callback(hObject, eventdata, handles)

    pos = hObject.Value;
    handles.TE_Registros.String = round(pos,0);
    cla (handles.Axes);

    if ~hObject.Value
        handles.TE_Texto_Registro.String = '';
        set(findall(handles.Panel_Registros,'Enable','On'),'Enable','Off');
        handles.PB_Nuevo_Registro.Enable = 'On';
        handles.T_Registro.Enable = 'On';
        handles.TE_Registros.Enable = 'On';
        handles.SL_Registros.Enable = 'On';
        return;
    end

    handles.TE_Texto_Registro.String = handles.Registros{pos}.Texto;


    if ~isempty(handles.Registros{pos}.data_imagen)
        imshow (imread( (handles.Registros{pos}.data_imagen)));
    end

    handles.SL_Hora_Registro.Value = str2double(handles.Registros{pos}.Hora);
    handles.TE_Hora_Registro.String = num2str(handles.Registros{pos}.Hora,'%02u');

    handles.SL_Minuto_Registro.Value = str2double(handles.Registros{pos}.Minuto);
    handles.TE_Minuto_Registro.String =
num2str(handles.Registros{pos}.Minuto,'%02u');

    handles.SL_Dia_Registro.Value = str2double(handles.Registros{pos}.Dia);
    handles.TE_Dia_Registro.String = num2str(handles.Registros{pos}.Dia,'%02u');

    handles.SL_Mes_Registro.Value = str2double(handles.Registros{pos}.Mes);
    handles.TE_Mes_Registro.String = num2str(handles.Registros{pos}.Mes,'%02u');

    handles.SL_Anio_Registro.Value = str2double(handles.Registros{pos}.Anio);
    handles.TE_Anio_Registro.String = num2str(handles.Registros{pos}.Anio,'%04u');

    set(findall(handles.Panel_Registros,'Enable','Off'),'Enable','On');


guidata(hObject,handles)

% --- Executes on button press in PB_Importar_Imagen.
function PB_Importar_Imagen_Callback(hObject, eventdata, handles)

    pos = str2double(handles.TE_Registros.String);

    image_path = uipickfiles('FilterSpec', [handles.AppFolder...
                 '\Proyectos\' handles.nombre_proyecto '\Imagenes\*.jpg'],...
                 'Prompt', 'Seleccione un archivo JPG', 'NumFiles', 1);

    if (~iscell(image_path) && ~image_path) return; end

    if (isempty(strfind(image_path{1},[handles.AppFolder...
        '\Proyectos\' handles.nombre_proyecto '\Imagenes\'])))

        p = 1;
        while( exist([handles.AppFolder '\Proyectos\'...
                 handles.nombre_proyecto '\Imagenes\Tracing_'...
                 handles.nombre_archivo num2str(p) '.jpg'],'file'))
            p = p + 1;
```

```
        end
        copyfile(image_path{1},[handles.AppFolder '\Proyectos\'...
                 handles.nombre_proyecto '\Imagenes\Tracing_'...
                 handles.nombre_archivo num2str(p) '.jpg']);
        image_path{1} = [handles.AppFolder '\Proyectos\'...
                 handles.nombre_proyecto '\Imagenes\Tracing_'...
                 handles.nombre_archivo num2str(p) '.jpg'];
    end

%     handles.Registros{pos}.data_imagen = imread( image_path{1} );
%     axes(handles.Axes);
%     cla (handles.Axes);
%     handles.Imagen = imshow (handles.Registros{pos}.data_imagen);

    handles.Registros{pos}.data_imagen = image_path{1};
    axes(handles.Axes);
    cla (handles.Axes);
    imshow ( imread( handles.Registros{pos}.data_imagen) );
    axis image;

guidata(hObject,handles);


function TE_Texto_Registro_Callback(hObject, eventdata, handles)
    pos = str2double(handles.TE_Registros.String);
    handles.Registros{pos}.Texto = hObject.String;
guidata(hObject, handles);

% --- Executes on button press in PB_Eliminar_Registro.
function PB_Eliminar_Registro_Callback(hObject, eventdata, handles)

    pos = str2double(handles.TE_Registros.String);
    if ~pos return; end

    handles.Registros(pos) = [];

    handles.SL_Registros.Value = ...
        min(handles.SL_Registros.Value, length(handles.Registros));

    handles.SL_Registros.Max = length(handles.Registros);
    if ~isempty(handles.Registros)
        handles.SL_Registros.SliderStep = [1 1].*1/(length(handles.Registros));
    else
        handles.SL_Registros.SliderStep = [1 1].*0;
    end

    SL_Registros_Callback(handles.SL_Registros, eventdata, handles);
    handles=guidata(hObject);

guidata(hObject,handles)

% --- Executes on button press in PB_Terminar_Edicion.
function PB_Terminar_Edicion_Callback(hObject, eventdata, handles)
    for i = 1:length(handles.Registros)
        if ~isempty(handles.Registros{i}.Plot)
            handles.Registros{i}.Plot.Visible = 'On';
            handles.Registros{i}.TextPlot.Visible = 'On';
        end
    end

    set(findall(handles.Panel_Registros,'Enable','On'),'Enable','Off');
    set(findall(handles.Panel_Menu_Trazo,'Enable','Of'),'Enable','On');
    set(findall(handles.Panel_Menu,'Enable','Of'),'Enable','On');
guidata(hObject, handles);


%Panel Menu
% --- Executes on button press in PB_Guardar.
function PB_Guardar_Callback(hObject, eventdata, handles)
% hObject    handle to PB_Guardar (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
    savefig(handles.output,[handles.path_proyecto '\Diary\' ...
            handles.nombre_archivo '.fig']);

guidata(hObject,handles);



    % --- Executes on button press in PB_PDF.
function PB_PDF_Callback(hObject, eventdata, handles)

    TextDatos1 = [];

    TextDatos1 = [TextDatos1 'Nombre del Proyecto: ' handles.nombre_proyecto '\n'];
    TextDatos1 = [TextDatos1 'Nombre del Archivo: ' handles.nombre_archivo '\n\n'];
    TextDatos1 = [TextDatos1 'Lugar de estudio: ' handles.lugar_estudio '\n'];
    TextDatos1 = [TextDatos1 'Nombre del Usuario: ' handles.nombre_usuario '\n\n'];

    TextDatos2 = [];
    TextDatos2 = [TextDatos2 'Condiciones climáticas: ' handles.Clima '\n'];
    TextDatos2 = [TextDatos2 'Fecha: ' handles.TE_Dia.String '/'
handles.TE_Mes.String '/' handles.TE_Anio.String '\n'];
    TextDatos2 = [TextDatos2 'Hora: ' handles.TE_Hora.String ':'
handles.TE_Minuto.String '\n\n'];


    f = figure('Units', 'Pixels', 'Position', [200 30 210 300].*2);


    uicontrol(f,'Style','Text','Units', 'Pixels', 'Position', [10,280,190,10].*2,...
    'String', ['Speed Test - ' handles.nombre_archivo], 'FontSize', 12,
'BackgroundColor','w');

    t = uicontrol(f,'Style','Text','Units', 'Pixels', 'Position',
[10,240,90,30].*2,...
    'FontSize', 6,  'HorizontalAlignment', 'Left', 'BackgroundColor','w');

    t.String = textwrap(t,{sprintf(TextDatos1)});

    t = uicontrol(f,'Style','Text','Units', 'Pixels', 'Position',
[110,240,90,30].*2,...
    'FontSize', 6,  'HorizontalAlignment', 'Left', 'BackgroundColor','w');

    t.String = textwrap(t,{sprintf(TextDatos2)});

    t(1) = uicontrol(f,'Style','Text','Units', 'Pixels', 'Position',
[10,165,90,65].*2,...
    'FontSize', 6,  'HorizontalAlignment', 'Left', 'BackgroundColor','w');

    t(2) = uicontrol(f,'Style','Text','Units', 'Pixels', 'Position',
[10,90,90,65].*2,...
    'FontSize', 6,  'HorizontalAlignment', 'Left', 'BackgroundColor','w');

    t(3) = uicontrol(f,'Style','Text','Units', 'Pixels', 'Position',
[10,15,90,65].*2,...
    'FontSize', 6,  'HorizontalAlignment', 'Left', 'BackgroundColor','w');

    h(1) = axes('Units', 'Pixels', 'Position', [110, 165, 90,65].*2, 'Visible',
'Off');
    h(2) = axes('Units', 'Pixels', 'Position', [110, 90, 90,65].*2, 'Visible',
'Off');
    h(3) = axes('Units', 'Pixels', 'Position', [110, 15, 90,65].*2, 'Visible',
'Off');

    Fpos = 1;
    while( exist([handles.path_proyecto '\Diary\PDF\' handles.nombre_archivo
num2str(Fpos) ' - 1.PDF'],'file'))
        Fpos = Fpos + 1;
    end

    numFile = 1;
```

```
    for i = 1: length(handles.Registros)

        T = handles.Registros{i};
        TextRegistro = [];

        TextRegistro = [TextRegistro 'Registro: ' num2str(i) '\n'];
        TextRegistro = [TextRegistro 'Fecha: ' T.Dia '/' T.Mes '/' T.Anio '\n'];
        TextRegistro = [TextRegistro 'Hora: ' T.Hora ':' T.Minuto '\n\n'];

        [r,~] = size(T.Texto);
        for j=1: r
            TextRegistro = [ TextRegistro strtrim(T.Texto(j,:)) '\n'];
        end

        pos = mod(i,3);
        if ~pos pos = 3; end

        t(pos).String = textwrap(t(pos),{sprintf(TextRegistro)});

        if ~isempty(T.data_imagen)
            h(pos).Visible = 'On';
            axes(h(pos));
            handles.Imagen = imshow (T.data_imagen);
        else
            h(pos).Visible = 'Off';
        end

        if( pos == 3 || i == length(handles.Registros) )


            figureName = [handles.path_proyecto '\Diary\PDF\'...
            handles.nombre_archivo num2str(Fpos) ' - ' num2str(numFile) '.PDF'];
            if exist(figureName,'file')
                delete(figureName)
            end

            set(gcf,'paperunits','centimeters');
            set(gcf,'papersize',[21 29.7]);
            set(gcf,'paperposition',[0,0,21,29.7]);
            print(gcf, '-dpdf',figureName);
            if ispc
                winopen(figureName);
            end
            numFile = numFile + 1;

            t(1).String = [];
            t(2).String = [];
            t(3).String = [];
            cla(h(1));
            cla(h(2));
            cla(h(3));

        end

    end

close(f);


% --- Executes on button press in PB_Cerrar.
function PB_Cerrar_Callback(hObject, eventdata, handles)

    choice = questdlg(sprintf(['¿Desea guardar el archivo antes de cerrarlo?']), ...
                              'Cerrar archivo', ...
                              'Guardar','Continuar','Guardar');
    if strcmp(choice,'Guardar')
            PB_Guardar_Callback(handles.PB_Guardar,eventdata,handles);
    end

    openfig([handles.path_proyecto '\Pantalla_Principal.fig']);

delete(handles.output);
```