Hawk application

The Hawk application has been created for fast demo creation and for research purpose. It represents to be a C interpreter (compatible with ISO C), with built in functions of OpenCV, IPL and interface functions to Win32.

Getting started

print a message

Type
printf("Hello Y%dK", 2);
in the editor (right pane), save the document and press Run button (or F5). In the trace window (left pane) you will see the printed message.
Note: it is not necessary to include stdio.h. The standard headers are included automatically.

Simple windows functionality. User input.

In the editor window type:
named_window("Y2K window", 0);
wait_key(0);
destroy_window("Y2K window");
Save the document. Run the script by pressing Run button or F5. A window with the specified name should be created. Make this window active and press any key. The window should be destroyed.

Work with IPL.

IPLIMAGE image;
image = load_iplimage("Y2K.bmp");
named_window("Y2K window", 0);
show_iplimage("Y2K window", image);
wait_key(0);

Hints

- Press the right mouse button in editor window to display the menu of modules. Select the module to display the list of available functions.
- Move the cursor to the function name to get a syntax hint on a status bar.
- Press the "Watch" button on the toolbar to quick watch variable values during script execution (use wait_key function to pause the execution).

Hawk reference

The application structure

The Hawk consists of the following parts and modules:

- common plugins structure
  - OpenCV that can be used in scripting
  - IPL that can be used in scripting
- The visualization library that can be used in scripting. The library represents a set of functions that make easy the use of Win32. It includes: creation of a window, showing a bitmap in a window, waiting for a keystroke.
- Video library
- The C interpreter CVEiCL. It is created on the basis of EiC, a C interpreter that is distributed under ARTISTIC licence and can be downloaded from www.anarchos.com/eic. In addition to standard runtime library, CVEiCL contains built in libraries OpenCV, IPL and HighGUI.
- The Hawk application itself that brings all the pieces together.

Common plugins structure

Hawk plugin is a dll module that is placed in the *plugins* directory and implements *Hawk plugin interface*, which is described below:

```
void SetErrLevel(errlevel_t);
```

errlevel_t declared in **hawkwrap.h** takes on values *safe* and *unsafe*. This function should set the safe mode, in which a long jump should be performed in case of fatal errors (see below). Is implemented in **eicdllwrap.h**.

```
void SetErrMark(jmp_buf*);
```

This function should set the mark for the long jump that should be performed in safe mode in case of fatal errors. Is implemented in **eicdllwrap.h**. The code in long jump should be a pointer to the dynamically allocated structure wrapexcept_t, declared in **hawkwrap.h.**

```
void SetEiCStack(AR_t**);
```

This function is intended to tune the pointer to the interpreter stack. It is implemented in **eicdllwrap.h**.

```
void SetEiCCallback(void(*)(void*));
```

This function is intended to tune the pointer to the EiC method that implements interpreters callbacks . It is implemented in **eicdllwrap.h**.

```
void GetFunctions(char***, val_t (***)(void), int*);
```

This function returnes pointers to the array of names and addresses of wrappers that are implemented in the module. Below is the description of two plugins, OpenCV and IPL.

OpenCV & IPL

The basic data unit for OpenCV and IPL is a pointer to IplImage structure, IPLIMAGE:

typedef iplImage* IPLIMAGE;

See Getting started for the examples of using IPL.

All IPL functions are accessible in the scripting environment.

Visualization

The visualization library is a set of interface functions to Win32. They enable to create a window, to show an image in a window without writing callback functions and to manage messages.

int named_window(const char* name, UINT flags);
int destroy_window(const char* name);
IPLIMAGE load_iplimage(const char* filename);
int save_iplimage(const char* filename, IPLIMAGE image);
int show_iplimage(const char* name, IPLIMAGE image);
int wait_key(const char* name);
int create_toolbar(const char* name);
int create_slider(const char* slider_name, const char* window_name, int* val, int count, void (*on_notify)(int));

Window styles
Error codes

named_window:

int named_window(const char* name, UINT flags);
Creates a simple window with the specified name. If the window with this name already exists, the function does nothing and reports an error.
Parameters:

name	window name. In HighGUI, every window is identified and referenced to by its name.
flags	HighGUI styles that should be set. See HighGUI window styles for a list of possible values.

Return value:
0 if the function succeeded, nonzero value in case of a failure.

destroy_window:

int destroy_window(const char* name);
Destroys a HighGUI window with the specified name. If the window with this name is not found, the function reports an error.
Parameters:

name	the name of the window to be destroyed.

Return value:
0 if the function succeeded, nonzero value in case of a failure.

load_iplimage:

IPLIMAGE load_iplimage(const char* filename);
Loads a bitmap from a Windows bitmap file and stores it in an IplImage structure.
Parameters:

filename	the name of the file.

Return value:
a valid IPLIMAGE pointer if succeeded, NULL in case of a failure.

save_iplimage:

int save_iplimage (const char* filename, IPLIMAGE image);
Saves an IplImage to a Windows bitmap file.
Parameters:

filename — the name of the file. If the filename contains no path, the file is attempted to be written first to *bitmaps* directory in the folder that contains Hawk.exe and then to ..\..*bitmaps* directory (the last option is added for developers with Hawk source codes).

image — the pointer to IplImage structure.

Return value:

HG_OK if succeeded or error code.

show_iplimage:

int show_iplimage(const char* name, IPLIMAGE image);

Displays the bitmap in the window. Also resizes the window to fit the bitmap.

Parameters:

name — the name of the window

image — the pointer to IplImage.

Return value:

0 if succeeded, nonzero in case of a failure.

wait_key:

int wait_key(const char* name);

Waits for a key press in a window with the specified name. If name is NULL, then waits for a key press in any of HighGUI windows.

Parameters:

name — should be a name of a HighGUI window or NULL

Return value:

The key code or 0 if the window that waits for the key press is being destroyed.

create_toolbar:

int create_toolbar(const char* name);

Creates an empty toolbar in the window with the specified name.

Parameters:

name — should be a name of a HighGUI window.

Return value:

HG_OK if succeeded, error code otherwise.

create_slider:

int create_slider(const char* slider_name, const char* window_name, int* val, int count, void (*on_notify)(int));

Creates a trackbar and places it on the toolbar. If the toolbar is not created up to the moment with the create_toolbar function, it is created automatically before trackbar creation.

Parameters:

name — should be a name of a HighGUI window

val — the address of the variable that will receive the trackbar position update. If a new trackbar is created then its position is initialized with the *val value; if the trackbar with the specified name already exists then, on the contrary, the *val receives the position. Can be NULL if the trackbar position is not needed.

count — ticks count of the slider; minimum and maximum positions are 0 and count correspondingly

void (*on_notify)(int) — notify function that will be called on each slider position change. Can be NULL if callbacks are not needed

Return value:

HG_OK if succeeded, error code otherwise.

HighGUI window styles
Here is the list of HighGUI window styles:

- HG_AUTOSIZE: The window with this style cannot be resized with the UI; it self adjusts the size to the image attached to it.

HighGUI error codes

- HG_OK: No comments
- HG_BADNAME: Is returned if the argument is char* and the string it points to is considered to be wrong by any means.
- HG_INITFAILED: Initialization of HighGUI library has failed. The most probable causes are that either the HighGUI window class registration had failed or the CvlGrFmts.dll library loading had failed.
- HG_WCFAILED: Returned if the window could not be created.
- HG_NULLPTR: Returned if the input pointer value is null and it should not.

Video library

play_avi:

int play_avi(char* filename, char* windowname, void (*callback)(IPLIMAGE));
Plays an avi file calling a script function on each frame. Returns when the video ends or if the user destroys the render window. The file path can be absolute. If it is relative then the file is searched in the windows system directories, then in the PATH, then in the Video directory on the same level as the Hawk.exe and finally in the Video directory two levels higher than Hawk.exe (the last is intended for work with sources in Microsoft Developer Studio).
Parameters:

filename	the name of an avi file
windowname	the name of the HighGUI render window where to render the video. If the window does not exist then it is created before and deleted after rendering.
callback	the function that will be called on each frame before rendering. Can be zero.

Return value:
zero if succeeded, nonzero in case of a failure.

play_ds:

int play_ds(void (*callback)(IPLIMAGE));
Starts DirectShow. Unlike running DirectShow with “PlayDS” button (see above) this method allows to start video processing after some initialization. Stop video with “Stop DS” button.
Parameters:

callback	the function that will be called on each frame before rendering. Can be zero.

Return value:
zero if succeeded, nonzero in case of a failure.

CVEiCL

The CVEiCL module is based on the ISO C interpreter EiC. It’s source codes were slightly modified to suit the needs of Hawk and thus renamed to CVEiCL. The OpenCV, IPL HighGUI and video modules are attached to CVEiCL in order to enable the calls to Win32 and image processing functions. The script should be written as a standard body of main() function. The script should be started with variables declarations. As usual, declarations after the first statement are not possible. The script is checked for errors before execution starts, so if it does not fit the ISO C grammar, the interpreter generates an error text message and does not execute the script.

Hawk

Basically, there are two ways to run scripts. The simple one is to press the Run button or F5 to simple execute it. Also you can press RunDS button to start DirectShow pipeline.

Simple execution
Hawk executes *.c files by interpreting a preprocessor command
#include "filename.c"
So the user should save the script to the file before executing it.
If you work with simple script execution you might need to load source images from files. You could do that with HighGUI method load_iplimage. It will search for the image first in the current directory, then in the **bitmaps directory**, which is placed in the same folder as **Hawk.exe**[1].

DirectShow execution
Every frame from you source will be processed with the script before being rendered. The frame could be referred to as ds_image declared as:
IPLIMAGE ds_image;
But it is not possible to apply the changes in your script during DirectShow session. The script is compiled to interpreter byte-code only once, before DirectShow starts. To change the script it is necessary to stop DirectShow.

Troubleshooting

- **The execution of the script is aborted with the following message: "Signal with code 2 has been raised by EiC in filename, linenumber. Aborting script execution."**
  - ✓ The script contains fatal errors like reading from null pointers. Check the variables that are referenced in the reported line.
  - ✓ Make sure that the function that is called from the reported line does exist.
  - ✓ Pay attention to the message following the abort message box (if any).
- **The button that runs DirectShow is disabled:**
  - ✓ Make sure that the DirectX Media 6.0 or compatible is installed correctly on your computer.
  - ✓ Make sure that no other application uses DirectShow.
  - ✓ Register the DirectShow filter ProxyTrans.ax with the following command:
    regsvr32 ProxyTrans.ax
- **The execution of the script is aborted and the script persistently fails afterwards**
  - ✓ Restart the application

[1] If the file with the specified name is not found in these folders, the application attempts to load it from ..\..\bitmaps directory, which is appropriate if the user works with source codes.