作业三-190410102-方尧

Halcon 程序

```
* Image Acquisition 01: Code generated by Image Acquisition 01
open framegrabber ('DirectShow', 1, 1, 0, 0, 0, 0, 'default', 8, 'rgb',
-1, 'false', 'default', '[0] XiaoMi USB 2.0 Webcam', 0, -1, AcqHandle)

grab image start (AcqHandle, -1)
while (true)
    grab image async (Image, AcqHandle, -1)
    * Image Acquisition 01: Do something
    write image( Image , 'bmp' , 0 , 'C:/Users/hebuyong/Desktop/1' )
endwhile
close_framegrabber (AcqHandle)
```

结果（1.bmp）

Cpp 程序

```
//////////////////////////////////////////////////////////////////////////
//////////
// File generated by HDevelop for HALCON/C++ Version 19.11.0.0
// Non-ASCII strings in this file are encoded in local-8-bit encoding
(cp936).
// Ensure that the interface encoding is set to locale encoding by calling
// SetHcppInterfaceStringEncodingIsUtf8(false) at the beginning of the
program.
//
// Please note that non-ASCII characters in string constants are exported
// as octal codes in order to guarantee that the strings are correctly
// created on all systems, independent on any compiler settings.
//
// Source files with different encoding should not be mixed in one project.
//////////////////////////////////////////////////////////////////////////
//////////



#ifndef   APPLE
#  include "HalconCpp.h"
#  include "HDevThread.h"
#else
#  ifndef HC_LARGE_IMAGES
#    include <HALCONCpp/HalconCpp.h>
#    include <HALCONCpp/HDevThread.h>
#    include <HALCON/HpThread.h>
#  else
```

```
#    include <HALCONCppxl/HalconCpp.h>
#    include <HALCONCppxl/HDevThread.h>
#    include <HALCONxl/HpThread.h>
#  endif
#  include <stdio.h>
#  include <CoreFoundation/CFRunLoop.h>
#endif



using namespace HalconCpp;


#ifndef NO EXPORT MAIN
// Main procedure
void action()
{

  // Local iconic variables
  HObject  ho Image;

  // Local control variables
  HTuple  hv AcqHandle;

  //Image Acquisition 01: Code generated by Image Acquisition 01
  OpenFramegrabber("DirectShow", 1, 1, 0, 0, 0, 0, "default", 8, "rgb",
-1, "false",
      "default", "[0] XiaoMi USB 2.0 Webcam", 0, -1, &hv AcqHandle);

  GrabImageStart(hv AcqHandle, -1);
  while (0 != 1)
  {
    GrabImageAsync(&ho Image, hv AcqHandle, -1);
    //Image Acquisition 01: Do something
    WriteImage(ho Image, "bmp", 0, "C:/Users/hebuyong/Desktop/1");
  }
  CloseFramegrabber(hv AcqHandle);




}


#ifndef NO EXPORT APP MAIN

#ifdef   APPLE
// On OS X systems, we must have a CFRunLoop running on the main thread
in
// order for the HALCON graphics operators to work correctly, and run the
// action function in a separate thread. A CFRunLoopTimer is used to make
sure
// the action function is not called before the CFRunLoop is running.
// Note that starting with macOS 10.12, the run loop may be stopped when
a
// window is closed, so we need to put the call to CFRunLoopRun() into
a loop
// of its own.
```

```
HTuple     gStartMutex;
H pthread t gActionThread;
HBOOL      gTerminate = FALSE;

static void timer callback(CFRunLoopTimerRef timer, void *info)
{
  UnlockMutex(gStartMutex);
}

static Herror apple action(void **parameters)
{
  // Wait until the timer has fired to start processing.
  LockMutex(gStartMutex);
  UnlockMutex(gStartMutex);

  try
  {
    action();
  }
  catch (HException &exception)
  {
    fprintf(stderr,"  Error #%u in %s: %s\n", exception.ErrorCode(),
           (const char *)exception.ProcName(),
           (const char *)exception.ErrorMessage());
  }

  // Tell the main thread to terminate itself.
  LockMutex(gStartMutex);
  gTerminate = TRUE;
  UnlockMutex(gStartMutex);
  CFRunLoopStop(CFRunLoopGetMain());
  return H MSG OK;
}

static int apple main(int argc, char *argv[])
{
  Herror              error;
  CFRunLoopTimerRef    Timer;
  CFRunLoopTimerContext TimerContext = { 0, 0, 0, 0, 0 };

  CreateMutex("type","sleep",&gStartMutex);
  LockMutex(gStartMutex);

  error = HpThreadHandleAlloc(&gActionThread);
  if (H MSG OK != error)
  {
    fprintf(stderr,"HpThreadHandleAlloc failed: %d\n", error);
    exit(1);
  }

  error = HpThreadCreate(gActionThread,0,apple action);
  if (H MSG OK != error)
  {
    fprintf(stderr,"HpThreadCreate failed: %d\n", error);
    exit(1);
  }

  Timer = CFRunLoopTimerCreate(kCFAllocatorDefault,
```

```
                              CFAbsoluteTimeGetCurrent(),0,0,0,
                              timer callback,&TimerContext);
  if (!Timer)
  {
    fprintf(stderr,"CFRunLoopTimerCreate failed\n");
    exit(1);
  }

CFRunLoopAddTimer(CFRunLoopGetCurrent(),Timer,kCFRunLoopCommonModes);

  for (;;)
  {
    HBOOL terminate;

    CFRunLoopRun();

    LockMutex(gStartMutex);
    terminate = gTerminate;
    UnlockMutex(gStartMutex);

    if (terminate)
      break;
  }


CFRunLoopRemoveTimer(CFRunLoopGetCurrent(),Timer,kCFRunLoopCommonMode
s);
  CFRelease(Timer);

  error = HpThreadHandleFree(gActionThread);
  if (H MSG OK != error)
  {
    fprintf(stderr,"HpThreadHandleFree failed: %d\n", error);
    exit(1);
  }

  ClearMutex(gStartMutex);
  return 0;
}
#endif

int main(int argc, char *argv[])
{
  int ret = 0;

  try
  {
#if defined( WIN32)
    SetSystem("use window thread", "true");
#endif

    // file was stored with local-8-bit encoding
    //   -> set the interface encoding accordingly
    SetHcppInterfaceStringEncodingIsUtf8(false);

    // Default settings used in HDevelop (can be omitted)
    SetSystem("width", 512);
    SetSystem("height", 512);
```

```
#ifndef   APPLE
    action();
#else
    ret = apple main(argc,argv);
#endif
  }
  catch (HException &exception)
  {
    fprintf(stderr,"  Error #%u in %s: %s\n", exception.ErrorCode(),
           (const char *)exception.ProcName(),
           (const char *)exception.ErrorMessage());
    ret = 1;
  }
  return ret;
}

#endif


#endif
```