```cpp
/*
 hhs_cam.cpp

 Get frames from DaHeng USB3.0 camera and
 display them in window
 Created on: 2022 / 07
 Author: Fidelis Theinert
 Reading DaHeng cameras with OpenCV 4.5
 Version 1.0
 */

#include <iostream>
#include <string>
#include <stdio.h>

#include <opencv.hpp>
#include <highgui.hpp>

#include "dh0.h"

// Namespace for using cout.
using namespace std;

// Namespace for OpenCV
using namespace cv;


/*****************************************************************************
**

 DEFINITIONS AND MACROS

 *****************************************************************************
*/

// blue green red is order used in openCV
#define COL_BLUE                    0
#define COL_GREEN                   1
#define COL_RED                     2

#define COLMODE_COL                 0
#define COLMODE_GREY                1

/*****************************************************************************
**

 PROTOTYPES OF NOT EXPORTED FUNCTIONS

 *****************************************************************************
*/

//int imgSave(int cnt, Mat img, string fname);
int Config(int argc, char **argv, struct ImgConf *camCfg);
int PrintHelp(void);
void InitWindows(string);
void ConvertGrey(Mat*);

/*****************************************************************************
```

```
**

 PROTOTYPES OF EXPORTED FUNCTIONS

 ****************************************************************************
*/

/****************************************************************************
**

 DEFINITIONS OF GLOBALS

 ****************************************************************************
*/

int ShowFPS = true;
int DisplayMode = COLMODE_COL;

struct ImgConf {
  int resolution;
  int camMode;
  double exposureMS;
};

//
****************************************************************************
int main(int argc, char *argv[]) {
//
****************************************************************************
  double t;
  int key;
  int cntframe = 0;

  string camName;
  char countxt[90];

  struct ImgConf cfg;

  // set default values
  cfg.resolution = CAM_RES_640_480;
  cfg.camMode = CAM_MODE_COL;
  cfg.exposureMS = 12.34; // setting default exposure time in milliseconds

  //  set the configuration according to commandline-parameters
  Config(argc, argv, &cfg);

  //  call the constructor and open default camera
  //  if this does NOT succeed the program will abort here (see: constructor)
  dh cam0(0);

  //  declare the matrix where our image is stored
  Mat image;

  //  set camera-mode and exposure-time
  cam0.setMode(cfg.resolution, cfg.camMode);
  cam0.setExpoMs(cfg.exposureMS);

  //  get camera name
  cam0.getName(&camName);
```

```
  cout << "using device '" << camName << "' " << endl;

  //  initialize our OpenCV display window
  InitWindows(camName);

  // discard first image to let camera settle
  cam0.captureFrame(&image);

  //  get systemtime to calculate frame-rate later on
  t = (double) getTickCount();

  // get actual exposuretime
  cam0.getExpoMs (&cfg.exposureMS);
  cout << "Using resolution: " << image.cols << " by " << image.rows
      << ", exposuretime: " << cfg.exposureMS << " ms" << endl;

  //  here the main-loop starts, read one frame
  while (cam0.captureFrame(&image) == CAM_OK) {
    //  increment frame counter
    cntframe++;

    //  check if retrieving image was successful
    if (!image.empty()) {

      //  check is we have to convert the image to grey-scale
      switch (DisplayMode) {
      case COLMODE_COL: // normal color
        break;

      case COLMODE_GREY: // grey-scale
        ConvertGrey(&image);
        break;
      }

      //  check if we have to display frame-rate
      if (ShowFPS == true) {
        //  define location where to display the frame-rate
        Point org;
        org.x = 10;
        org.y = 30;

        //  calculate the expired time since last acquisition of frame
        t = ((double) getTickCount() - t) / getTickFrequency();

        sprintf(countxt, "fps: %4.1f [%06d], exp: %6.2f [ms]",
            (1.0 / t), cntframe, cfg.exposureMS);
//        sprintf(countxt, "fps: %4.1f [%06d], exp: %6.2f [ms]",
//            (1.0 / t), cntframe, cam0.getExpoMs());

        //  get new time
        t = (double) getTickCount();

        //  print string to image-buffer
        putText(image, countxt, org, 1, 2, Scalar(0, 255, 255), 2, 16,
            false);
      }

      // display frame in standard window
```

```
      imshow(camName, image);
    }

    // make frame visible
    key = waitKey(1);

//    if (key != -1)
//      cout << "key: '" << key << "'  " << endl;

// check for 'Esc' (or 'backspace' or 'enter') to stop
    if ((key == 0x1b) || (key == 0x08) || (key == 0x0d)) {
      cout << "Stopping Cam!" << endl;
      cam0.close();
      break;
    } else {
      // check for keyboard commands
      switch (key) {

      case '?':
        cout << "ROI width = " << image.cols << ", height = "
            << image.rows << endl;
        break;

      case 'e':
        cout << "Exposure time set to: " << cfg.exposureMS << " ms"
            << endl;
        break;

      case ' ':
        // save image using openCV API
        break;
      }
    }
  }

  return 0;
}

//
*****************************************************************************
void ConvertGrey(Mat *image) {
//
*****************************************************************************
  //  go through all cols and rows and convert each pixel to gray value
  //  grey = 0.299 * red + 0.587 * green + 0.114 * blue
  for (int r = 0; r < image->rows; r++) {
    for (int c = 0; c < image->cols; c++) {
      Vec3b &rgb = image->at<Vec3b>(r, c);

      rgb[COL_RED] = (unsigned char) (0.299 * (float) rgb[COL_RED]
          + 0.587 * (float) rgb[COL_GREEN]
          + 0.114 * (float) rgb[COL_BLUE]);
      rgb[COL_GREEN] = rgb[COL_RED];
      rgb[COL_BLUE] = rgb[COL_RED];
    }
  }
}

//
```

```
//
//*************************************************************************
int Config(int argc, char **argv, struct ImgConf *camCfg) {
//
//*************************************************************************
  //  read commandline-parameters one by one
  if (argc > 1) {
    for (int i = 1; i < argc; i++) {
      if (argv[i][0] == '-') {

        //  check for help
        if (argv[i][1] == '?') {
          PrintHelp();
        }

        //  check for frames per second display
        if (argv[i][1] == 'F') {
          cout << "show FPS!" << endl;
          ShowFPS = true;
        }

        //  check for grey-scale display
        if (argv[i][1] == 'G') {
          cout << "show grey-scale image" << endl;
          DisplayMode = COLMODE_GREY;
        }
      }
    }
  } else {
    PrintHelp();
  }

  return 0;
}

//
//*************************************************************************
int PrintHelp(void) {
//
//*************************************************************************
  cout << "DaHeng USB3 Camera-Framework, V1.0" << endl;
  cout << "(c) F. Theinert 2022" << endl;
  cout << "Commandline options: -F -G -?" << endl;
  cout << "  -F show frames per second" << endl;
  cout << "  -G grey-scale image" << endl;
  cout << "  -? this help-screen" << endl;

  return 0;
}

//
//*************************************************************************
void InitWindows(string camName) {
//
//*************************************************************************

  // make HighGui OpenCV window for display
  namedWindow(camName, WINDOW_AUTOSIZE | WINDOW_GUI_NORMAL);
}
```

```
275
276 ///* EOF hhs_cam.cpp  */
277
```