VCodec interface C++ library

v2.1.4

Table of contents

Overview

VCodec C++ library provides standard interface as well defines data structures and rules for different video codec classes (video encoding and decoding). **VCodec** interface class doesn't do anything, just provides interface. Different video codec classes inherit interface form **VCodec** C++ class. **VCodec.h** file contains **VCodecCommand** enum, **VCodecParam** enum and **VCodec** class declaration. **VCodecCommands** enum contains IDs of commands supported by **VCodec** class. **VCodecParam** enum contains IDs of params supported by **VCodec** class. All video codec should include params and commands listed in **VCodec.h** file. **VCodec** class depends on Frame class which determines video frame structure. Video codec interface supports only 8 bit depth input pixels. It uses C++17 standard. The library is licensed under the **Apache 2.0** license.

Versions

Table 1 - Library versions.

Version	Release date	What's new
1.0.0	14.06.2023	First version.
1.1.0	20.06.2023	- Added new parameter.
1.1.1	28.06.2023	- Frame submodule updated. - Documentation updated. - License added. - Repository made public. - Added new parameters.
1.1.2	02.08.2023	- Frame submodule updated.
2.1.1	14.12.2023	- Virtual destructor added. - Frame submodule updated.
2.1.2	14.12.2023	- Frame submodule updated. - Documentation updated.
2.1.3	25.04.2024	- Documentation updated.
2.1.4	16.05.2024	- Documentation updated. - Frame submodule updated.

Library files

The library supplied by source code only. The user would be given a set of files in the form of a CMake project (repository). The repository structure is shown below:

```
CMakeLists.txt ---------- Main CMake file of the library.
3rdparty ---------------- Folder with third-party libraries.
    CMakeLists.txt ------ CMake file to include third-party libraries.
    Frame --------------- Folder with Frame library files.
src --------------------- Folder with library source code.
    CMakeLists.txt ------ CMake file of the library.
    VCodec.h ------------ Main library header file.
    VCodecVersion.h ----- Header file with library version.
    VCodecVersion.h.in -- File for CMake to generate version header.
    VCodec.cpp ---------- C++ implementation file.
```

Video codec interface class description

Class declaration

VCodec interface class declared in **VCodec.h** file. Class declaration:

```
class VCodec
{
public:

    /// Class destructor.
    virtual ~VCodec();

    /// Get string of current library version.
    static std::string getVersion();

    /// Encode/decode.
    virtual bool transcode(Frame& src, Frame& dst) = 0;

    /// Set parameter.
    virtual bool setParam(VCodecParam id, float value) = 0;

    /// Get parameter.
    virtual float getParam(VCodecParam id) = 0;

    /// Execute command.
    virtual bool executeCommand(VCodecCommand id) = 0;
};
```

getVersion method

The **getVersion()** method returns string of current version of **VCodec** class. Particular video codec class can have it's own **getVersion()** method. Method declaration:

```
static std::string getVersion();
```

Method can be used without **VCodec** class instance:

```
std::cout << "VCodec class version: " << VCodec::getVersion() << std::endl;
```

Console output:

```
VCodec class version: 2.1.4
```

transcode method

The **transcode(...)** method intended to encode and decode video frame (Frame class). Video codec encodes / decodes video frame-by-frame. Method declaration:

```
virtual bool transcode(Frame& src, Frame& dst) = 0;
```

Parameter	Value
src	Source video frame (see Frame class description). To encode video data **src** frame must have RAW pixel data (field **fourcc** of **Frame** class): **RGB24**, **BGR24**, **YUYV**, **UYVY**, **GRAY**, **YUV24**, **NV12** (default format for codec), **NV21**, **YU12** or **YV12**. To decode video data **src** frame must have compressed pixel format (field **fourcc** of **Frame** class): **JPEG**, **H264** or **HEVC**. Particular video codec can support limited RAW input pixel format or only one (usually only **NV12**). When it possible video codec should accept all supported RAW pixel formats and should do pixel format conversion inside if it necessary. Also, particular video codec can support all, few or just one compressed pixel format. When it possible video code should support all compressed pixel format to encode / decode.
dst	Result video frame (see Frame class description). To decode video data **src** frame must have compressed pixel format (field **fourcc** of **Frame** class): **JPEG**, **H264** or **HEVC**. In case decoding particular video codec can set output pixel format automatically. To encode video frame user must set **fourcc** field of dst frame to necessary output compressed format: **JPEG**, **H264** or **HEVC**. The method will write decoded frame data (RAW pixel format) to **data** filed of **src** frame in case decoding or will write compressed data in case encoding.

Returns: TRUE if frame was encoded/decoded or FALSE if not.

setParam method

The **setParam(...)** method designed to set new video codec parameters value. Method declaration:

```
virtual bool setParam(VCodecParam id, float value) = 0;
```

Parameter	Description
id	Video codec parameter ID according to VCodecParam enum.
value	Video codec parameter value.

Returns: TRUE is the parameter was set or FALSE if not.

getParam method

The **getParam(...)** method designed to obtain video codec parameter value. Method declaration:

```
virtual float getParam(VCodecParam id) = 0;
```

Parameter	Description
id	Video codec parameter ID according to VCodecParam enum.

Returns: parameter value or -1 of the parameters doesn't exist in particular video codec class.

executeCommand method

The **executeCommand(...)** method designed to execute video codec command. Method declaration:

```
virtual bool executeCommand(VCodecCommand id) = 0;
```

Parameter	Description
id	Video codec command ID according to VCodecCommand enum.

Returns: TRUE is the command was executed or FALSE if not.

Data structures

VCodec.h file defines IDs for parameters (**VCodecParam** enum) and IDs for commands (**VCodecCommand** enum).

VCodecCommand enum

Enum declaration:

```
enum class VCodecCommand
{
    /// Reset.
    RESET = 1,
    /// Generate key frame. For H264 and H265 codecs.
    MAKE_KEY_FRAME
};
```

Table 2 - Video codec commands description. Some commands maybe unsupported by particular video codec class.

Command	Description
RESET	Reset video codec.

Command	Description
MAKE_KEY_FRAME	Command to generate key frame for H264 or H265(HEVC) encoding.

VCodecParam enum

Enum declaration:

```
enum class VCodecParam
{
    /// [read/write] Log level:
    /// 0-Disable, 1-Console, 2-File, 3-Console and file.
    LOG_LEVEL = 1,
    /// [read/write] Bitrate, kbps. For H264 and H265 codecs.
    BITRATE_KBPS,
    /// [read/write] Quality 0-100%. For JPEG codecs.
    QUALITY,
    /// [read/write] FPS. For H264 and H265 codecs.
    FPS,
    /// [read/write] GOP size. For H264 and H265 codecs.
    GOP,
    /// [read/write] H264 profile: 0 - Baseline, 1 - Main, 2 - High.
    H264_PROFILE,
    /// [read/write] Codec type. Depends on implementation.
    TYPE,
    /// Custom 1. Depends on implementation.
    CUSTOM_1,
    /// Custom 2. Depends on implementation.
    CUSTOM_2,
    /// Custom 3. Depends on implementation.
    CUSTOM_3
};
```

Table 3 - Video codec params description. Some params maybe unsupported by particular video codec class.

Parameter	Access	Description
LOG_LEVEL	read / write	Logging mode. Default values: 0 - Disable. 1 - Only file. 2 - Only terminal. 3 - File and terminal.
BITRATE_KBPS	read / write	Bitrate, kbps. For H264 and H265(HEVC) encoding. According to this value, FPS and GOP size video codec calculate parameter for H264 or H265(HEVC) encoding.
QUALITY	read / write	Quality 0(low quality)-100%(maximum quality). For JPEG encoding.

Parameter	Access	Description
FPS	read / write	FPS. For H264 and H265 codecs. According to this value, FPS and GOP size video codec calculate parameter for H264 or H265(HEVC) encoding.
GOP	read / write	GOP size (Period of key frames) for H264 or H265(HEVC) encoding. Value: 1 - each output frame is key frame, 20 - each 20th frame is key frame etc.
H264_PROFILE	read / write	H264 profile for H264 encoding: 0 - Baseline, 1 - Main, 2 - High.
TYPE	read / write	Codec type. Depends on implementation. It can be type of backend. Some codecs may not support this parameter.
CUSTOM_1	read / write	Custom parameter. Depends on particular implementation.
CUSTOM_2	read / write	Custom parameter. Depends on particular implementation.
CUSTOM_3	read / write	Custom parameter. Depends on particular implementation.

Build and connect to your project

Typical commands to build **VCodec** library:

```
git clone https://github.com/ConstantRobotics-Ltd/VCodec.git
cd VCodec
git submodule update --init --recursive
mkdir build
cd build
cmake ..
make
```

If you want connect **VCodec** library to your CMake project as source code you can make follow. For example, if your repository has structure:

```
CMakeLists.txt
src
    CMakeList.txt
    yourLib.h
    yourLib.cpp
```

You can add repository **VCodec** as submodule by commands:

```
cd <your respository folder>
git submodule add https://github.com/ConstantRobotics-Ltd/VCodec.git 3rdparty/VCodec
git submodule update --init --recursive
```

In you repository folder will be created folder **3rdparty/VCodec** which contains files of **VCodec** repository with subrepositories **Frame**. New structure of your repository:

```
CMakeLists.txt
src
    CMakeList.txt
    yourLib.h
    yourLib.cpp
3rdparty
    VCodec
```

Create CMakeLists.txt file in **3rdparty** folder. CMakeLists.txt should contain:

```
cmake_minimum_required(VERSION 3.13)

###############################################################################
## 3RD-PARTY
## dependencies for the project
###############################################################################
project(3rdparty LANGUAGES CXX)

###############################################################################
## SETTINGS
## basic 3rd-party settings before use
###############################################################################
# To inherit the top-level architecture when the project is used as a submodule.
SET(PARENT ${PARENT}_YOUR_PROJECT_3RDPARTY)
# Disable self-overwriting of parameters inside included subdirectories.
SET(${PARENT}_SUBMODULE_CACHE_OVERWRITE OFF CACHE BOOL "" FORCE)

###############################################################################
## CONFIGURATION
## 3rd-party submodules configuration
###############################################################################
SET(${PARENT}_SUBMODULE_VCODEC                          ON  CACHE BOOL "" FORCE)
if (${PARENT}_SUBMODULE_VCODEC)
    SET(${PARENT}_VCODEC                                ON  CACHE BOOL "" FORCE)
endif()

###############################################################################
## INCLUDING SUBDIRECTORIES
## Adding subdirectories according to the 3rd-party configuration
###############################################################################
if (${PARENT}_SUBMODULE_VCODEC)
    add_subdirectory(VCodec)
endif()
```

File **3rdparty/CMakeLists.txt** adds folder **VCodec** to your project. Your repository new structure will be:

```
CMakeLists.txt
src
    CMakeList.txt
    yourLib.h
    yourLib.cpp
3rdparty
    CMakeLists.txt
    VCodec
```

Next you need include folder 3rdparty in main **CMakeLists.txt** file of your repository. Add string at the end of your main **CMakeLists.txt**:

```
add_subdirectory(3rdparty)
```

Next you have to include VCodec library in your **src/CMakeLists.txt** file:

```
target_link_libraries(${PROJECT_NAME} VCodec)
```

Done!