VCodec interface C++ library

v1.1.0

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.

Versions

Table 1 - Library versions.

Version	Release date	What's new
1.0.0	14.06.2023	First version.

Version	Release date	What's new
1.1.0	20.06.2023	- Added new parameter.

Video codec interface class description

Class declaration

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

```
namespace cr
{
namespace video
{
/**
 * @brief Video codec interface class.
 */
class VCodec
{
public:
    /**
     * @brief Get string of current library version.
     * @return String of current library version.
     */
    static std::string getVersion();

    /**
     * @brief Get new video frame.
     * @param src Source frame (RAW or compressed).
     * @param dst Result frame (RAW or compressed).
     * @return TRUE if frame was processed or FLASE if not.
     */
    virtual bool transcode(Frame& src, Frame& dst) = 0;

    /**
     * @brief Set video codec param.
     * @param id Parameter ID.
     * @param value Parameter value to set.
     * @return TRUE if parameter was set of FALSE.
     */
    virtual bool setParam(VCodecParam id, float value) = 0;

    /**
     * @brief Get video codec parameter value.
     * @param id Parameter ID according to camera specification.
     * @return Parameter value or -1.
     */
    virtual float getParam(VCodecParam id) = 0;

    /**
     * @brief Execute command.
     * @param id Command ID .
     * @return TRUE if the command accepted or FALSE if not.
     */
```

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

getVersion method

getVersion() method return 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;
```

transcode method

transcode(...) method intended to encode and decode video frame (**Frame** class). Video codec encode/decode video frames 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**, **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. 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**.

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

setParam method

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

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

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.
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
};
```

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.
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.

Parameter	Access	Description
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.